Diseño e implementación de una arquitectura de integración basada en la WoT para una "Smart Factory"

Acuña Piñán, Néstor Iván

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

Ing. Alulema Flores, Darwin Omar, PhD.

7 de julio del 2022

COPYLEAKS

Trabajo_Titulacion_Acuna_Pinan_Nestor_Ivan.pdf

Scanned on: 0:36 July 10, 2022 UTC

DARWIN OMAR ALULEMA FLORES

1.2%	9	22562
Overall Similarity Score	Results Found	Total Words in Text

Identical Words	134
Words with Minor Changes	39
Paraphrased Words	89
Omitted Words	0

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Certificación

Certifico que el trabajo de titulación: "**Diseño e implementación de una arquitectura de integración basada en la WoT para una "Smart Factory"**" fue realizado por el señor **Acuña Piñán, Néstor Iván;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 07 de julio del 2022

Firma:

DARWIN OMAR
ALULEMA
FLORES

.....

Dr. Alulema Flores, Darwin Omar

C.C: 1002493334

Departamento de Eléctrica, Electrónica y Telecomunicaciones

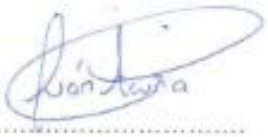
Carrera de Ingeniería en Electrónica, Automatización y Control

Responsabilidad de Autoría

Yo, **Acuña Piñán, Néstor Iván**, con cédula/cédulas de ciudadanía n°1725282238, declaro que el contenido, ideas y criterios del trabajo de titulación: "**Diseño e implementación de una arquitectura de integración basada en la WoT para una "Smart Factory"**", es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 07 de julio del 2022

Firma



.....

Acuña Piñán, Néstor Iván

C.C.: 1725282238

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingenieria en Electrónica, Automatización y Control

Autorización de Publicación

Yo, **Acuña Piñán, Néstor Iván** con cédula de ciudadanía n°1725282238, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**Diseño e implementación de una arquitectura de integración basada en la WoT para una "Smart Factory""**, en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 07 de julio del 2022

Firma

Acuña Piñán, Néstor Iván

C.C.: 1725282238

Dedicatoria

Esta tesis está dedicada a:

A dios nuestro señor por permitirme culminar la meta profesional más importante de mi vida hasta el momento, nunca desampararme, señor nunca me faltes sin tu infinito amor no sería posible cumplir este sueño.

A Ney Paladines y Alexandra Acuña dedico a ustedes todo mi trayecto universitario y el presente trabajo de titulación, por confiar y creer en mí de manera incondicional, me faltará vida para poder recompensarles todo lo que hicieron por mí.

A mi madre y padre por su amor, paciencia, consejos y apoyo ante todas las adversidades que se presentaron en este largo camino.

Para ustedes, ¡Lo logramos!

Agradecimiento

Un camino lleno de adversidades, triunfos, derrotas y sobre todo personas que estuvieron dispuestas a ayudarme, que Dios las puso en mi camino, así describiría esta etapa llena de también alegrías y triunfos como ahora.

Agradezco en primer lugar a Dios, mi consuelo en todos los momentos difíciles, mi guía, mi luz, solo tú me conoces y conoces mis intenciones, solo puedo decirte gracias, por poner en mi camino a personas que me impulsaron a soñar, el sueño que tuve de niño se cumplió.

A mis tíos Ney y Alexandra por apoyarme todos estos años desde el primer día hasta hoy, porque a pesar de la distancia los sentía alado, por su apoyo incondicional, por estar en la etapa que muchas personas me dieron la espalda, por confiar en mis capacidades, por brindarme las herramientas necesarias, lo que nadie más daría por mí. Por su apoyo, aliento, incentivo y amor les agradezco de corazón, la vida me permitirá recompensarles con el doble, por permitirme vivir este sueño.

Gracias madre por tu amor único, por tu motivación y jugarte la vida día a día con la esperanza de algún día poder vivir este sueño, nuestro sueño.

Gracias padre por forjarme en carácter y apoyarme a seguir esta carrera una pequeña decisión que pudo cambiar mi vida.

Finalmente agradezco a cada uno de mis familiares y amigos que me apoyaron y supieron estar ahí, por su apoyo incondicional. A la universidad y sus docentes y sobre todo a mi tutor Ing. Darwin Alulema por su paciencia y conocimientos brindarme las pautas y ser el artífice principal del desarrollo del presente proyecto.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

El desarrollo tecnológico de la cuarta revolución industrial se encuentra en auge, todos los avances apuntan hacia el uso de servicios web, componentes ciberfísicos e internet de las cosas como medio. Grandes industrias centran sus esfuerzos en la convergencia de su área operativa con las tecnologías de la información para obtener los beneficios que una industria 4.0 otorga. Mediante esta motivación surge la propuesta de una herramienta de software para impulsar el desarrollo productivo e investigativo de esta área. La arquitectura propuesta está basada en de tres capas: Física, Lógica y de Aplicación basándose en la Web Of Things la cual es una especialización del Internet de las Cosas. Se conceptualiza como componentes web a los sensores y actuadores de la fábrica para mediante su orquestación puedan ser coordinarlos y realizar el control y monitoreo de estos en la nube. Este proceso anteriormente se lo realizaba de manera local y los componentes no podían interactuar para un fin común. Al estar en la nube se realiza una fácil integración a servidores remotos como el correo electrónico y más herramientas que son esenciales para una industria 4.0. Cabe recalcar que los sensores y actuadores son simulaciones de sus mecanismos mediante servicios web y que no están implementados. Finalmente se realiza un escenario de prueba el cual es una etapa de una fábrica embotelladora para comprobar el correcto funcionamiento de la arquitectura y está visualizado mediante un Front-End en Ignition. Para validar la arquitectura obtendremos resultados de pruebas de carga, usabilidad y eficiencia con la arquitectura propuesta.

Palabras clave: Industria 4.0, Web de las Cosas, Sistemas ciberfísicos, fábrica inteligente.

Abstract

The technological development of the fourth industrial revolution is booming, all advances point towards the use of web services, cyber-physical components and the Internet of things as a medium. Large industries focus their efforts on the convergence of their operational area with information technologies to obtain the benefits that an industry 4.0 grants. Through this motivation arises the proposal of a software tool to promote the productive and investigative development of this area. The proposed software is based on a three-layer architecture: Physical, Logical and Application based on the Web Of Things which is a specialization of the Internet of Things. The sensors and actuators of the factory are conceptualized as web components so that, through their orchestration, they can be coordinated and controlled and monitored in the cloud. This process was previously carried out locally and the components could not interact for a common purpose. Being in the cloud allows for easy integration with remote servers such as email and more tools that are essential for an industry 4.0. It should be noted that the sensors and actuators are simulations of their mechanisms through web services and that they are not implemented. Finally, a test scenario is carried out, which is a stage of a bottling factory to check the correct functioning of the architecture and is visualized through a Front-End in Ignition. To validate the architecture, we will obtain load, usability and efficiency test results with the proposed architecture.

Key words: Industry 4.0, Web Of Things, Cyber-physical systems, Smart Factory.

Capítulo 1: Marco Metodológico

Antecedentes

La Industria 4.0 es la cuarta revolución industrial, y supone un cambio, abrupto en la forma de fabricación, logística y la organización del trabajo. Consiste en la digitalización de los procesos industriales, y está enfocada en intentar organizar todos los medios productivos y toda la empresa en sí, para conseguir mejores resultados teniendo al internet como base de interconexión y las implicaciones que esto supone en cuanto a la facilidad de acceso a la información. Las tecnologías y herramientas utilizadas en la Industria 4.0 son, por ejemplo, la realidad virtual y aumentada, el IoT (Internet of Things), inteligencia y visión artificial, asistentes virtuales, Big Data, cloud computing, programas modernos de diseño y de simulación de procesos, entre otras (ISOTools, 2018). A pesar de que el término de Industria 4.0 ha venido cobrando fuerza en los últimos años, no es nada más que una idea a implementarse globalmente en un futuro, por lo tanto, aún no existe un mecanismo estándar ni universal para conectar una industria 4.0, además el desarrollo se ha visto dificultado, debido a que la mayoría de empresas (principalmente las medianas y pequeñas) no cuentan con una automatización total de todos sus procesos industriales, y para que exista una Industria 4.0 es necesario que las máquinas y a veces las piezas o productos deben ser “inteligentes” (Hans, 2015).

Dentro de una fábrica inteligente el Sistema de Ejecución de la Producción (MES) controla las operaciones de producción, es decir, permite una conexión completa de producción, con el fin de manejar toda la información del proceso productivo en un sistema general con una infraestructura de integración y que sea eficaz. Por estas razones, las grandes industrias centran sus esfuerzos para que los sistemas de las Tecnologías de la Información (IT) y Tecnologías Operativas (OT) converjan de forma precisa en la capa MES dentro de la pirámide de automatización. Siendo los Sistemas de las Tecnologías de la Información (IT) principalmente utilizados para la computación centrada en datos y los Sistemas de Tecnología

Operacional (OT), utilizados para supervisar eventos, procesos y dispositivos, los cuales obtienen la información que la IT procesa. En cuanto a los beneficios que genera esta convergencia entre ambos sistemas, se debe destacar que es utilizada cantidad de datos que son generados por los sistemas OT, y a partir de la interpretación de estos datos, con estos beneficios se pretende que IT y OT tengan una gran escabilidad, uno de los requisito es que se pueda acceder en tiempo real a todo el contenido y a la información originada para conveniencia del operario (Cano, 2018).

Para que una Industria 4.0 sea posible todos los dispositivos deben interactuar entre sí y actualmente existe una limitación que se presenta cuando se intenta integrar dispositivos de varios fabricantes en una aplicación, dado que, cada fabricante utiliza su propio protocolo y por consiguiente se tiene cientos de protocolos incompatibles, haciendo que integrar los datos y servicios de varios dispositivos sea complejo y costoso. Para lograr que estos dispositivos se comuniquen entre sí, se necesita construir un protocolo de aplicación único y universal, es decir que todos los dispositivos hablen en un mismo lenguaje para entenderse, pero esto desafortunadamente es imposible en la actualidad, por este motivo una solución puede ser reutilizar la web, la cual es utilizada por millones de usuarios para crear aplicaciones que además pueden ser fácilmente escalables e interactivas. Dentro de este marco la Web of Things es una especialización de Internet de las Cosas, que consiste en reutilizar protocolos, estándares y demás funcionalidades que están disponibles y son populares, con los datos y servicios que ofrecen las Cosas y permitiría que cualquier dispositivo pueda integrarse y ser utilizado por cualquier aplicación, independientemente de los protocolos o estándares de red que ese dispositivo utilice (Guinard & Trifa, 2016).

El intercambio de datos e interoperar sistemas en la web, necesita que estos puedan estar basados en distintas tecnologías o que están alejados y lograr su comunicación con éxito. Para cumplir con este objetivo se utiliza el Web Service, es una interfaz que es accesible por

protocolos de red, que son usados en internet y permite acceder, insertar y modificar un objeto, mediante sus métodos, sin importar las tecnologías ni plataformas en que está dada la petición, también otorgan la ventaja de flexibilidad en procesos de empresas que permiten la integración de aplicaciones que caso contrario no podrían comunicarse, y actualmente se basan en estándares XML o JSON. En conclusión, un Web Service está encargado de compartir datos entre dos aplicaciones diferentes, y tiene la función abstraer el programa para que cualquier lenguaje de programación pueda entender los datos enviados. (Lequerica, 2003).

La forma actual de su implementación es por medio de microservicios ya que permiten una escalabilidad y mayor seguridad al momento de implementar sistemas web. Estas características son necesarias al integrar los sistemas de las tecnologías operativas (OT) con las tecnologías de la información (IT).

Los microservicios son un conjunto servicios que dividen una aplicación monolítica general en diferentes áreas. Son independientes e interactúan entre sí, cada uno cubre un área de la aplicación y pueden comunicarse mediante peticiones HTTP a sus API, debido a que son independientes pueden encontrarse en servidores diferentes y si un microservicio cae, los demás no se verían afectados y hasta pueden ser escritos en lenguajes de programación diferentes cada uno. Estos microservicios para poder realizar la función para la que fueron creados deben ser orquestados, es decir interactuar entre sí para realizar un fin común. Este tipo de coordinación de servicios, llamada orquestación, va a permitir establecer los mecanismos de control que pueden ser empleados en industrias para el diseño de los sistemas de control de los procesos, los cuales dejan de ejecutarse de forma local y pasan a ejecutarse en la nube, y al estar en la nube gozar de todas las facilidades que la web puede ofrecer y teniendo la capacidad de integrarse a otros servicios externos.

El presente proyecto pretende integrar las tecnologías web con el IoT con la finalidad de promover el desarrollo de la integración de las tecnologías de operación y las tecnologías de

la información, con el propósito de proveer mecanismos que faciliten la implementación de sistemas de coordinación y gestión en industrias o empresas enfocadas hacia la Industria 4.0.

Justificación e importancia

Debido al auge del Internet de las Cosas (IoT) es común encontrarse con definiciones acerca de lo que es una fábrica inteligente y la importancia que tiene no solo en la actualidad, sino a un futuro cercano. Uno de los objetivos que varias empresas tienen en común es dominar el futuro desarrollo y la producción de sistemas complejos. Para lo cual estos sistemas deben incorporar mayor flexibilidad y sobre todo productividad a su proceso en general, a través de la implementación y adaptación de la automatización industrial integrada en sus sistemas. Los cuales deben contener cada vez más sensores y capacidades de comunicación inalámbrica, con el fin de recopilar la información suficiente de datos y ser capaces de gestionar y analizar las grandes cantidades de datos. La Industria 4.0 está vinculada a la convergencia de IT y OT, esta convergencia genera una gran cantidad de datos que los sistemas OT generan; con una correcta interpretación de estos datos se generan beneficios en cualquier industria. Muchos procesos industriales pueden optimizarse para aumentar la eficiencia y facilitar así una toma de decisiones, más rápida y más rentable.

Los diversos aspectos en los que se presentan ventajas al implementar una Industria 4.0, son los siguientes:

- Medio ambiente: Controlar la contaminación que genera una industria dentro de los parámetros establecidos por la ley.
- Productivos: Adaptación según requerimientos del cliente, con una producción flexible y acorde a la demanda.
- Control de calidad: Fabricar con mayor calidad y sin aumentar el tiempo que esto conlleva.

- Mantenimiento: Realizar mantenimientos predictivos para reducir riesgo de averías y pérdidas de materia prima.

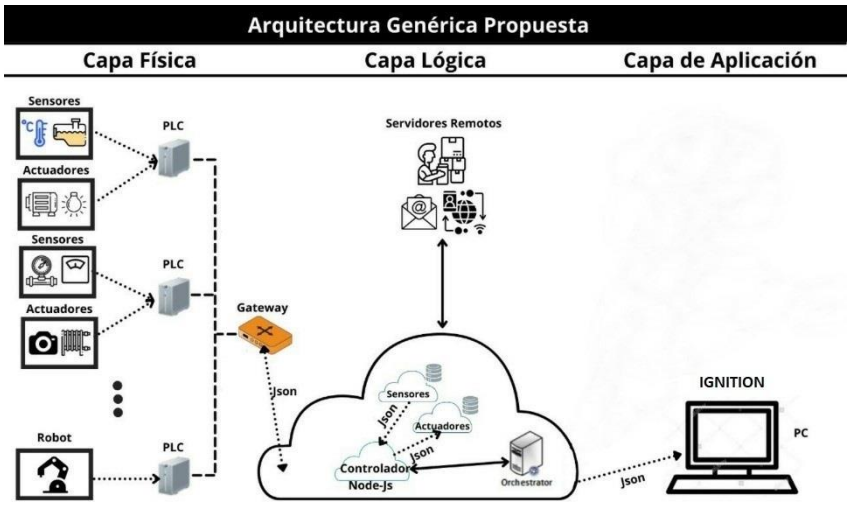
Debido a las falencias que presenta la Industria 3.0 asimilándose a un conjunto de máquinas aisladas, que no interactúan entre sí, ni se puede dar uso de la información que las tecnologías operativas generan y quedando claro los beneficios que una Industria 4.0 provee, este proyecto pretende crear una herramienta que permita la gestión de industrias inteligentes a partir de poder conceptualizar los componentes ciberfísicos como componentes web, los cuales puedan ser coordinados empleando técnicas de automatización para la implementación de técnicas industriales que a su vez pueden integrarse con facilidad hacia los sistemas de tecnologías de la información existentes en las industrias.

Alcance

Con el fin de promover el avance de la industria 4.0 y facilitar el desarrollo productivo, en este trabajo se propone una arquitectura basada en software que permite la automatización de industrias por medio de la implementación de servicios web, para mediante su orquestación coordinar los procesos productivos. De esta manera la propuesta es un modelo genérico de Smart Factory la cual es una adaptación de una arquitectura de tres capas: física, lógica y de aplicación.

Figura 1

Arquitectura genérica propuesta para una Smart Factory



Como se observa en la figura 1 la capa física está compuesta por: Nivel de Campo, es el encargado de adquisición de datos por parte de los sensores, el Nivel de Control o Supervisión aquí se encuentran los Controladores Lógicos Programables (PLC) los cuales reciben la información recopilada de todos los dispositivos del nivel inferior y a continuación, se tiene el Nivel SCADA, que para la arquitectura genérica propuesta es el Gateway o pasarela, que su principal función es la de enviar ordenes en tiempo real a uno o más PLC a la vez y en este caso en particular enviar o recibir información desde la nube. La Capa Lógica, comprende el Back-End del proyecto que se propone, tiene la principal función de procesar la información que fue enviada en formato JSON a través del Gateway; los sensores y actuadores son considerados como componentes web, los cuales son coordinados mediante orquestación, es decir en este nivel se realizaron las técnicas de control del proceso y se envían las respuestas del orquestador en el mismo formato JSON mediante el Gateway a los actuadores, a su vez al estar en la nube se puede interactuar con servidores remotos externos como el correo electrónico. La Capa de Aplicación, representa el Front-End, corresponde a una plataforma en

la cual permita desplegar de manera visual los atributos de cada componente web y el estado operativo de la industria, para este fin se eligió el software IGNITION el cual permite monitorear en tiempo real y agregar una orden de producción para comprobar el funcionamiento del proyecto.

Los objetos en formato JSON enviados y recibidos en la capa lógica contienen la descripción de las cosas ciberfísicas, donde se describen los metadatos y las interfaces de las cosas, en la cual una cosa es una abstracción de una entidad física o virtual que proporciona interacciones y participa en la Web de las Cosas. Por ejemplo, en la tabla 1 se observa la Thing Description de una banda transportadora la cual permitió construir el objeto Json que es procesado por los distintos componentes del sistema, esta misma idea se extiende a todos los componentes genéricos que contiene el modelo genérico de una Smart Factory.

Tabla 1

Descripción de atributos de una banda transportadora genérica

Cosa	Atributos/Descripción	
Banda Transportadora	**Tipo**	Rodillo
		Tornillo Sin Fin
		Suelo Móvil
	Modelo	Genérico
	Estado	Nuevo/Usado
		Mantenimiento (Variable de tiempo en la que necesita mantenimiento).
		Dañado
	Ubicación	Sucursal Sangolquí

Cosa	Atributos/Descripción	
		Posición GPS
	Modo de	Lógica (1 – Encendido / 0 – Apagado)
	operación	Velocidad (rpm)
	Observación	Ninguna

Recapitulando, dentro de la capa lógica se encuentra la orquestación, esta es una tecnología de operación encargada de la coordinación de los componentes ciber físicos, que son vistos como componentes web, y es realizada mediante un conjunto de instrucciones, en las cuales existen técnicas de control que dan el funcionamiento adecuado a la fábrica inteligente y su vinculación con los servicios remotos. En este caso se implementa un orquestador por medio de un servicio web, el cual es el encargado de recibir los datos necesarios de las “cosas” mediante datos tipo JSON para que este realice el control (orquestación) de los actuadores.

Se ha considerado un particular caso de estudio para poner a prueba la propuesta. Se escoge una fábrica embotelladora la cual posee cinco procesos como se observa en la figura 2, de los cuales se realiza el subproceso de envasado para comprobar el correcto funcionamiento de la arquitectura genérica planteada.

Figura 2

Procesos de una fábrica embotelladora

En la etapa de Envasado los sensores y actuadores involucrados fueron simulados por un servicio web que genera información y recibe información, es decir se realizó una virtualización de los sensores, no se implementó físicamente. Esta virtualización consiste en un servicio web que genera datos JSON, los cuales son simulaciones de todas las “cosas” definidas en la arquitectura del modelo genérico para demostrar su funcionamiento. Estos servicios web están coordinados en la nube con sus propios servicios web los cuales interactúan entre ellos para poder generar información y esta es desplegada en tiempo real mediante Ignition.

En el subproceso de envasado se tiene 3 estaciones, la primera estación simula el movimiento de las botellas mediante sensores de presencia, cilindros para detener las botellas y una banda transportadora para permitir el movimiento de las mismas, la segunda estación simula el llenado del líquido en el cual se emplearon sensores de nivel y cilindros neumáticos, los cuales ingresan las válvulas de llenado en las botellas y una tercera estación que coloca las tapas a las botellas previamente llenadas, mediante cilindros. Simulando de esta manera con las tres estaciones un subproceso que es el escenario de prueba. Para las pruebas se consideró modelos genéricos para sensores y actuadores, para lo cual se modelan sus atributos como instancias de objetos JSON. Para el esquema de control del proceso se considera un orquestador el cual está desplegado como un servicio web que coordina las acciones de los servicios asociados a los sensores y actuadores.

La visualización del proceso simulado es monitoreada en Ignition, mediante las herramientas que proporciona, se puede configurar parámetros de inicio del proceso tales como orden de producción, cantidad de botellas a entregar y la materia prima ingresada, la información mostrada son gráficos históricos de los componentes virtuales simulados.

Objetivos

General

Diseñar e implementar una arquitectura basada en la Web de las Cosas para la automatización e integración de tecnologías operativas con tecnologías de la información en fábricas inteligentes.

Específicos

- Investigar los conceptos teóricos de: SOA (Arquitectura Orientada a Servicios), WOT (Web de las Cosas), REST, CPS (Sistemas Ciber físicos), IT (Tecnologías de la Información), OT (Tecnologías Operativas), Microservicios, Web Service, Smart Factory (Fábrica Inteligente).
- Diseñar una arquitectura de integración basada en la WoT que permite la coordinación de componentes ciber físicos.
- Analizar los mecanismos de virtualización de componentes ciber físicos para su emulación e integración en sistemas basados en la web.
- Modelar la arquitectura de integración para el despliegue de los servicios de orquestación con los cuales se coordinan los servicios de los componentes virtuales.
- Analizar el funcionamiento del sistema basado en la arquitectura de integración propuesta, mediante un escenario simulado de una fábrica embotelladora inteligente en el subproceso de envasado.
- Evaluar el comportamiento del sistema por medio de pruebas de usabilidad del sistema y carga del sistema.

Estado del Arte

Metodología

Una vez justificado el proyecto y su alcance, y que la Web Of Things es la arquitectura base para el diseño e implementación del proyecto propuesto, es importante estudiar los trabajos realizados hasta la fecha, en torno al tipo de tecnologías que han sido usadas para construir arquitecturas para una Industria 4.0, y si es posible que sugieran posibles soluciones en el avance del proyecto, así mismo se busca conocer si este tipo de investigaciones se aplicaron a la actualidad y sus resultados. Por lo cual se opta por realizar un Mapeo Sistemático de la Literatura para conocer el avance a nivel mundial sobre el tema del proyecto y en que rincones del mundo está siendo investigado, también entrega la primicia de descubrir que partes no están siendo trabajadas y tal vez sean importantes tratarlas en el desarrollo del proyecto. De la misma manera se realiza también una revisión sistemática de la literatura, la cual ayuda adentrar en el tema para conocerlo a profundidad más centrado en específicamente el tema que se tiene en desarrollo (Moguel, 2018).

Mapeo Sistemático de la Literatura. Celaya et al. (2020) menciona que la metodologia de Systematic Mapping Study (SMS), permite indagar en un tema altamente amplio, y permite construir clasificaciones, categorizando publicaciones, tesis, revistas, papers, que permitan obtener la frecuencia, la fecha y dónde fueron escritos, permitiendo determinar la cobertura del tema que se está investigando.

1. Planteamiento de preguntas. Sabiendo que la industria 4.0 está en auge recientemente es posible que los estudios encontrados en los que se utilice la WoT para una Smart Factory sean tan solo unos cuantos, por lo tanto, los esfuerzos se centran en seguir alguna metodología para poder realizar el mapeo y aprovechar de todo estudio que mencione alguna parte clave del proyecto, cómo virtualizar los sensores, actuadores, conceptualizarlos como componentes web o cómo realizar su orquestación mediante un servicio web.

Siguiendo la metolodología de Kroll et al.(2018) primero se establecen las preguntas de investigación y son planteadas desglozando el objetivo central del proyecto. Estas preguntas son útiles debido a que como en el caso del proyecto presente permiten indagar en temas en torno a diseños e implementaciones de industria 4.0 (ó Smart Factory) con arquitecturas en base a la IoT y más especifico en la WoT, software que fueron utilizados, la manera de conceptualizar a los componentes ciberfísicos de la fábrica y la técnica con la que se coordinan esos componentes en la actualidad, las preguntas planteadas se observan en la tabla 2.

Tabla 2

Preguntas de investigación planteadas

Pregunta de investigación	Motivación
RQ1: ¿Qué tipo de propuestas de arquitecturas existen para diseñar o implementar una fábrica inteligente o industria 4.0?	Conocer que arquitecturas existen actualmente para industrias 4.0 y su avance.
RQ2:¿Qué tipo de técnicas se utilizan para interconectar una fábrica inteligente?	Conocer las técnicas existentes, protocolos y tipos de interconexion de los componentes de industrias 4.0.
RQ3:¿Que tecnologías existen para orquestar microservicios?	Conocer como se puede coordinar los microservicios de la fábrica.
RQ4:¿Qué tecnicas existen para implementar servicios web?	Conocer las formas en las que se puede implementar un servicio web.

Nota. Realizada en base a una tabla obtenida de (Moguel, 2018).

2. Cadenas de búsqueda. Definiendo el motor de búsqueda se utilizó la Web Of Science debido a que es una base de datos muy completa, además la universidad tiene un convenio con este servicio de información científica, por lo tanto se procede a generar las cadenas a

partir de los términos clave de las preguntas planteadas anteriormente, para ingresar a Web Of Science.

Tabla 3

Términos utilizados para la cadena de búsqueda

Término clave	Términos alternativos
RQ1: Fábrica Inteligente	“Smart Factory” OR “Industry 4.0”
RQ2: WoT	IoT OR WoT OR “Web of Things”
RQ3: Orquestar Microservicios	Orchestration OR Microservices
RQ4:Servicios Web	“Web Service” OR Rest

Nota. Realizada en base a una tabla obtenida de (Moguel, 2018).

Cadena 1. (“Smart Factory” OR “Industry 4.0”)
AND (IoT OR WoT OR "Web of Things")
AND (Orchestration OR Microservices)
AND (“Web Service” OR “Rest”)

Con la cadena 1, que es la cadena completa de búsqueda no se obtienen resultados, esto era de esperar, debido a lo dicho anteriormente que la industria 4.0 está en auge recientemente y a esto agregándole palabras claves como WoT o Web Services las cuales son palabras que definen específicamente el proyecto que se desea desarrollar, da de resultado cero publicaciones con la busqueda de la cadena completa.

Se opta entonces, por realizar 3 cadenas de búsqueda combinadas para poder abarcar todos los temas y tener los estudios suficientes para se pueda establecer resultados fiables, el criterio para crear las cadenas combinadas es por conveniencia.

Cadena 2. (“Smart Factory” OR “Industry 4.0”)
AND (IoT OR WoT OR "Web of Things")

AND (Orchestration OR Microservices)

Cadena 3. (“Smart Factory” OR “Industry 4.0”)

AND (Orchestration OR Microservices)

Cadena 4. (“Smart Factory” OR “Industry 4.0”)

AND (IoT OR WoT OR "Web of Things")

AND (“Web Services” OR Rest)

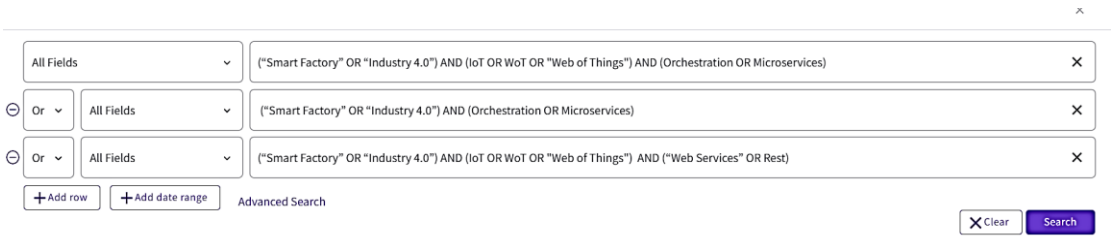
3. Criterios de inclusión y exclusión. Kitchenham et al. (2011) menciona, que se realiza un filtrado de la cantidad original de estudios para obtener una proporcion cantidad-calidad alta y que facilite la interpretación de los estudios encontrados.

Inclusión. Toda la Colección principal de Web of Science que va desde el año 2010 hasta el presente 2022, se aceptan todo tipo de documentos debido a que de origen el tema a buscar ya es reducido.

Exclusión. Unicamente se excluyen documentos duplicados o que no sean pertinentes a la rama de ingeniería a tratar, en caso hubiere y documentos de acceso cerrado.

Figura 3

Cadenas de búsqueda en Web Of Science



Nota. Captura tomada de la página de Web Of Science (Science, 2021).

Una vez se han aplicado las combinaciones de cadenas mencionadas anteriormente en Web Of Science como se observa en la figura 3, arrojó 64 resultados de los cuales 32 son de

libre acceso, no son necesarios más filtros debido a que la cantidad de estudios es aceptable para poder crear un mapa visual que facilite verificar los avances del desarrollo del tema que se está tratando.

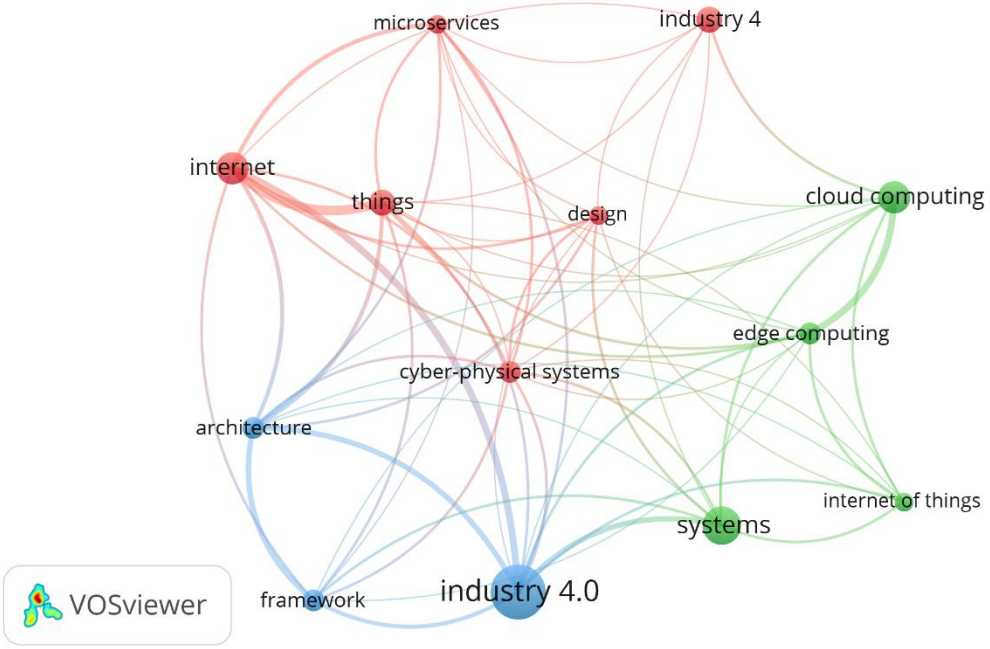
Los resultados son presentados mediante el software VOSviewer el cual es una herramienta gráfica que mediante redes bibliométricas permite analizar cientos de literaturas científicas, este software además permite analizar redes de cualquier tipo de estudio sea tesis, revista, artículo proveniente de cualquier base de datos como Scopus, Web Of Science, PubMed; estas redes de visualizacion científica tienen el beneficio de ilustrar contenido científico de una manera más facil al investigador (Waltman & Van Eck, 2013).

Además permite construir y visualizar graficamente relaciones como pueden ser, entre palabras clave o keywords de los estudios, así como los autores, países de los que proviene, revistas, organizaciones de investigación y condiciones específicas de cada una de las relaciones. Para crear una red se importa la lista de literaturas científicas entregada por las cadenas ingresadas en Web Of Science que previamente han sido filtradas y esa lista genera un archivo de texto plano, se importa ese archivo en VOSviewer y se elije el método de análisis que puede ser de tipo co-autorship, o co-occurrence, el primero permite crear la red en base a número de citaciones de autores, organizaciones o países y el segundo en base a número de repeticiones de palabras clave en la lista de documentos.

4. Resultados.

Figura 4

Red generada con palabras claves



Nota. Se generó esta red con el programa VOSviewer.

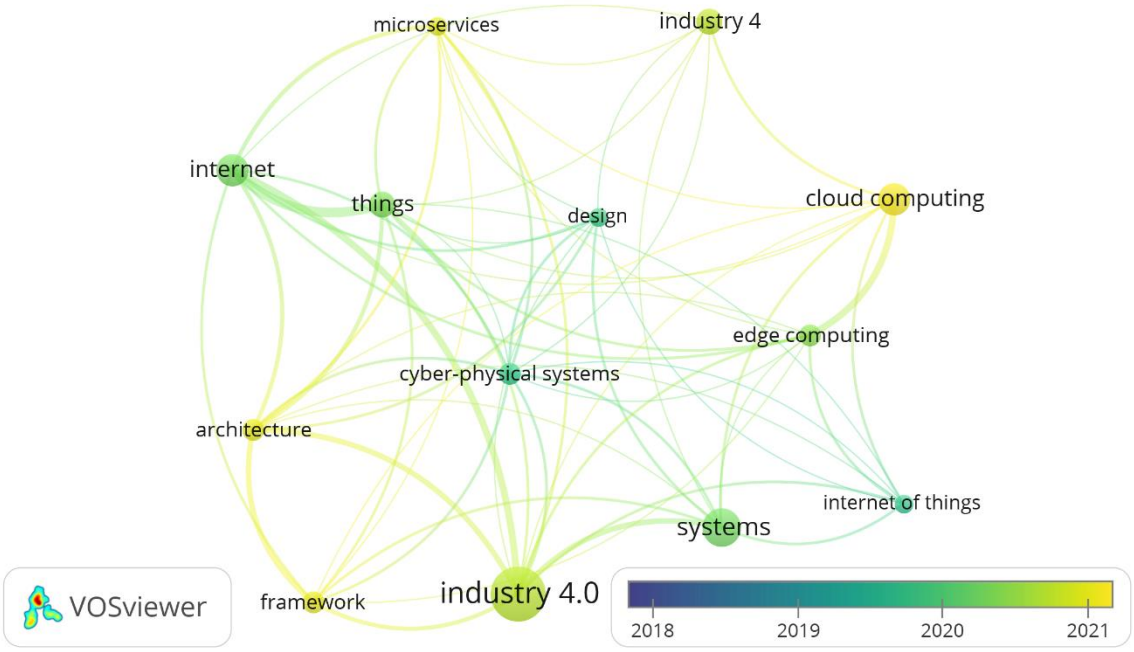
La red creada en VOSviewer se muestra en la figura 4, esta red a sido creada con el análisis de co-ocurrencia con las palabras clave de los documentos científicos, palabras que han sido un mínimo de 6 veces mencionadas en la lista de documentos proporcinada por Web of Science y en total fueron 13 palabras que cumplieron con este requisito, se puede observar según el tamaño del nodo, cuales fueron más mencionadas siendo industria 4.0 la más mencionada y la que conecta con mas nodos como: arquitectura, microservicios, sistemas ciberfísicos, cosas e internet de las cosas.

Se concluye entonces que una industria 4.0 tiene fuerte enlace con conceptos como los microservicios, las cosas, el internet, y los sistemas ciberfísicos en la lista de documentos que Web Of Science generó y en todos los estudios estas palabras convergen y muestran el

camino hacia al cual se está investigando, de esta manera se puede asegurar que la arquitectura que el proyecto propone es válida, debido a que también apunta a la dirección de investigación para una industria 4.0 que se ha venido investigando.

Figura 5

Red generada con palabras clave, en el tiempo

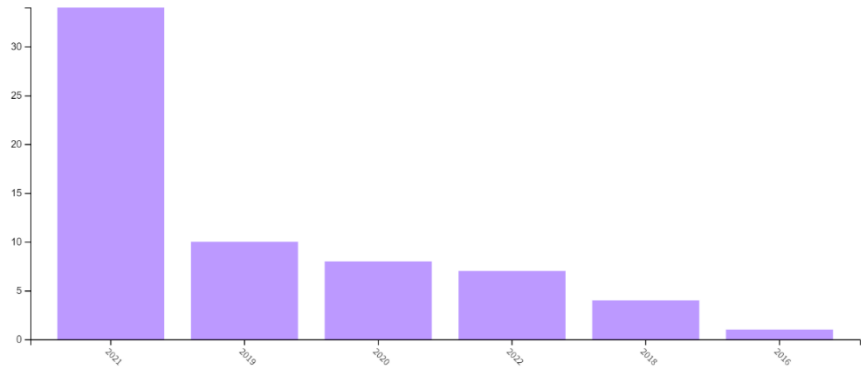


Nota. Se generó esta red con el programa VOSviewer.

Cambiando el modo de visualizacion a Overlay en VOSviewer se observa que los nodos de color más oscuro son las investigaciones más antiguas, como el IoT, los sistemas cyberfísicos y las cosas (things), pero no son tan antiguas pues son aproximadamente del 2019, y las más actuales son los microservicios el cloud computing, para aseverar los tiempos que indica VOSviewer, ya que el periodo de tiempo es muy corto tan solo de dos años se utiliza la herramienta de análisis de web of science.

Figura 6

Número de publicaciones de WoS en cada año

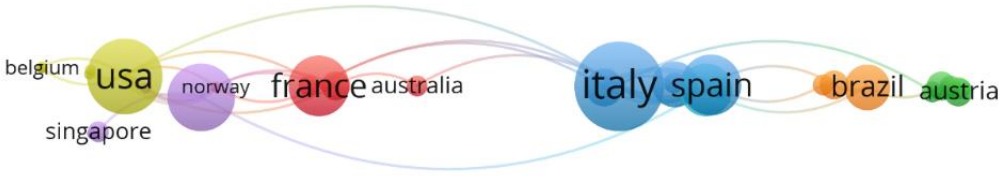


Nota. Gráfico proveniente del análisis del listado filtrado ingresado y es generado por Web of Science.

Se confirma la información entregada por VOSviewer en la figura 6, indicando que en el 2021 existieron alrededor de 40 publicaciones, 10 publicaciones el 2019 y el resto de los años tan solo unas cuantas, en conclusión el número de publicaciones a ascendido de manera exponencial en los ultimos 3 años, se denota interés en las industrias 4.0 y se investigan técnicas y tecnologias como las palabras clave mostradas en VOsviewer como los servicios web, microservicios, internet de las cosas, para hacer de la industria 4.0 una realidad.

Figura 7

Red generada con los países de las literaturas científicas



Nota. Se generó esta red con el programa VOSviewer

Finalmente se genera la red con los países de los que provienen los documentos científicos de la lista filtrada, para establecer si a nivel mundial se está trabajando en las líneas de desarrollo de la industria 4.0, como se observa en la figura 7, siendo Estados Unidos, Francia, Italia y España los de mayor publicaciones. Ecuador país del que proviene el presente proyecto, no registra una sola publicación según el listado, es por ello la importancia en la cual radica el proyecto la cual es promover la industria 4.0, mediante la generación de una arquitectura de software basada en la WoT.

Revisión Sistemática de la Literatura. Para completar los trabajos previos se realiza también una Systematic Literature Review (SLR), es una técnica de revisión que evalúa e identifica una lista de trabajos investigativos a través de un proceso sistemático, tiene como objetivo recopilar literaturas científicas, analizarlas y de esta manera se obtiene el Estado del Arte para el proyecto o área que se tenga como objetivo. La principal diferencia con el mapeo sistemático es que este aborda a profundidad un tema en específico, trata de una revisión más

depurada mientras un SMS aborda un tema de forma más general por lo tanto más amplia y las preguntas planteadas, así mismo, son menos descriptivas que una SLR (García, 2017).

Se utiliza a continuación una SLR en vista de que, las técnicas y tecnologías utilizadas para el diseño de una industria 4.0 están muy dispersas en los resultados obtenidos al realizar el SMS, y se desea conocer a profundidad que métodos se están trabajando para interconectar una Smart Factory, en dónde se realiza el control del proceso, sea en la nube o en los propios PLC y como repercute al desarrollo productivo, la tendencia actual y desafíos futuros.

La metodología para encontrar una SLR es muy similar a la utilizada anteriormente en el SMS a diferencia que las preguntas deben evitar resultados muy generales y se debe centrar en refinar estas preguntas lo más posible identificando las necesidades específicas de investigación.

1. Planteamiento de preguntas. Las preguntas planteadas se muestran en la tabla 4, las cuales tienen claro los objetivos que la SLR desea abordar.

Tabla 4

Planteamiento de preguntas para una SLR

Pregunta de investigación	Motivación
RQ1: ¿Qué tecnologías de interconexión como la WoT se utilizan en la industria 4.0?	Conocer tecnologias como la Web of Things, que sirvan de base para implementar una industria 4.0
RQ2:¿Qué tipos de orquestadores se utilizan en la coordinacion de microservicios?	Conocer actualmente que tipos de orquestadores se utilizan.
RQ4:¿Qué técnicas se utiliza para orquestar procesos industriales?	Conocer que técnicas se utilizan para coordinar procesos industriales.

Pregunta de investigación	Motivación
RQ3:¿Que tecnicas convierten componentes ciberfisicos en componentes web?	Conocer técnicas que conviertan componentes de la industria en componentes web.

2. Cadenas de búsqueda. Para la cadena de busqueda que se ingresa a Web Of Science se han definido los términos mostrados en la tabla 5, partiendo de las anteriores preguntas planteadas.

Tabla 5

Términos utilizados en la SLR

Término clave	Términos alternativos
RQ1: Web of Things	WoT OR “Web of Things”
RQ2: Orquestadores OR Microservicios	Orchestration OR Orchestrator OR Microservices
RQ3: Orquestador OR Procesos Industriales	Orchestration OR Orchestrator OR Control OR Industrial
RQ4:Ciberfisicos OR Componentes WEB	“Cyber-Physical” OR CPS OR “Components Web”

Una vez definidas las palabras clave se comprueba si existen resultados con la combinación completa de todas las palabras clave de todas las preguntas.

Cadena 1. (WoT OR “Web of Things”)
AND (Orchestration OR Orchestrator OR Microservices)
AND (Orchestration OR Orchestrator OR Control OR Industrial)

AND (“Cyber – Physical” OR CPS OR “Components Web”)

Para la cadena 1, nuevamente no existen resultados, por lo tanto, se vuelven a realizar cadenas combinadas.

Cadena 2. (WoT OR “Web of Things”)

AND (Orchestration OR Orchestrator OR Control OR Industrial)

AND (“Cyber – Physical” OR CPS OR “Components Web”)

Cadena 3. (WoT OR “Web of Things”)

AND (Orchestration OR Orchestrator OR Microservices)

AND (“Cyber – Physical” OR CPS OR “Components Web”)

Cadena 4. WoT OR “Web of Things”)

AND (Orchestration OR Orchestrator OR Control OR Industrial)

3. Criterios de inclusión y exclusión. Nuevamente se realiza un filtrado en la SLR, para los cuales los documentos deben cumplir los siguientes criterios.

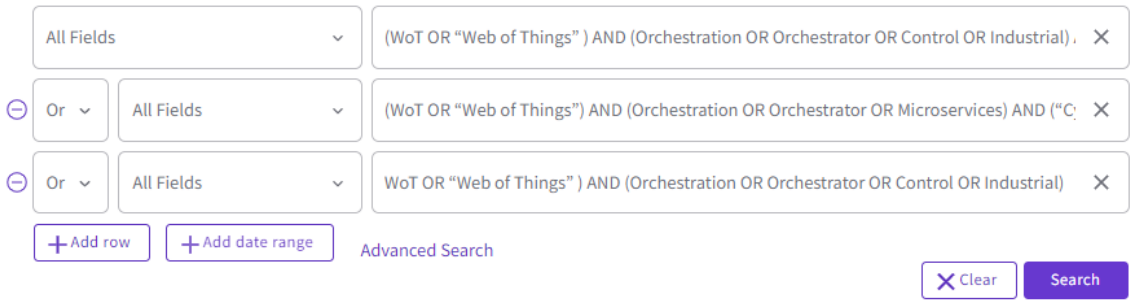
Inclusión. Toda literatura científica proveniente desde 2010 hasta la fecha, debido a que aún las investigaciones en el tema de la web of things son reducidas y se concluyó anteriormente con el SMS que una arquitectura con la web of things para una fábrica inteligente no se existe en la base de datos de web of science y al ser el SLR aún más especifico se aborda cualquier documento que pueda aportar.

Exclusión. Se excluyen documentos duplicados y que no pertenezcan al área abordada, estén incompletos o no se pueda tener acceso.

La cadena ingresada en web of science se muestra en la figura 8.

Figura 8

Cadena combinada ingresada a WoS

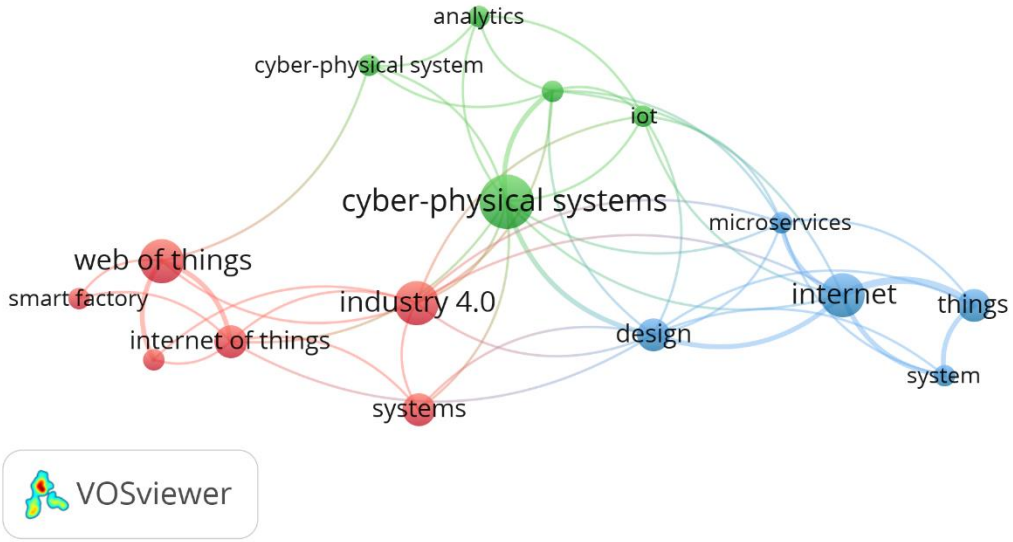


Nota. Captura tomada de la página de Web Of Science (Science, 2021).

4. Resultados.

Figura 9

Red con palabras clave para SLR

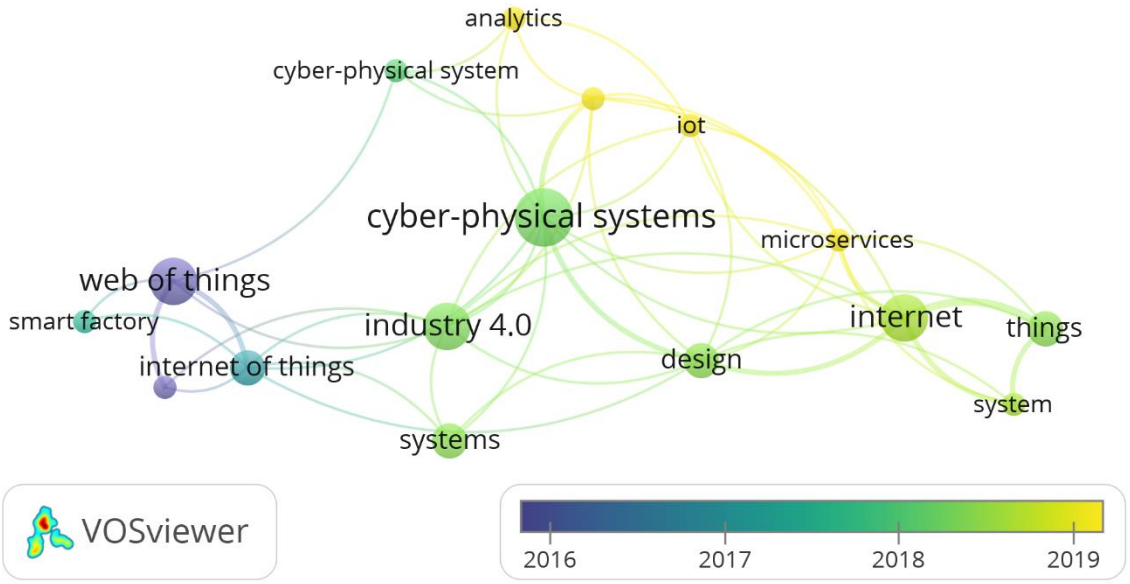


La figura 9 muestra la red en VOSviewer y fue creada aplicando el método de revisión sistemática de la literatura en el que se puede apreciar que el término más mencionado en los artículos es los sistemas ciber físicos, y tiene fuerte acoplamiento con la industria 4.0 el IoT, los

microservicios, por lo que indica que la técnica de implementar sistemas ciber físicos es muy estrecho al diseño de una fábrica 4.0 en los documentos científicos.

Figura 10

Red generada con palabras clave de SLR, en el tiempo



En la figura 10, se observa la red creada con las palabras clave según el año y que términos como la industria 4.0 los sistemas ciberfísicos pertenecen a 2018 aproximadamente, mientras que microservicios son recientes, los documentos que incluyen la palabra clave web of Things son de 2016 aproximadamente, el término Smart Factory esta entre 2017 y 2018. Se puede concluir que la mayor parte de estudios pertenecen al año 2018 y que está en avance el implementar la cuarta revolución en las industrias.

Figura 11

Red generada con países de los que provienen los documentos para LSR

Finalmente, en la figura 11 muestra los países que están trabajando con estos términos siendo Estados Unidos nuevamente el líder en investigación del avance de la industria, seguido por China e Italia. Esta red permite analizar en que partes del mundo se encuentran estudiando el área, y las debilidades actuales del sistema científico mundial en Latinoamérica.

Capítulo 2: Marco Conceptual

Tecnologías de la información (IT)

Los dos ejes de los que depende el presente proyecto son las tecnologías de la información y las tecnologías operativas, la convergencia entre estas permite el diseño de la arquitectura para una Smart Factory, las tecnologías de la información refieren a la utilización de equipos de telecomunicaciones los cuales involucran computadoras, softwares, servidores dedicados a almacenar, procesar, interpretar e intercambiar información e implementar servicios. Estos sistemas también incluyen protocolos mediante los cuales se permite la construcción de redes que brindan servicios en los cuales el medio de comunicación es el internet, permitiendo implementar el Internet de las Cosas el cual es clave para el diseño de la arquitectura de red de una fábrica la cual depende de la interconectividad, el traspaso de información y su análisis que puede ser utilizado de inmediato por cualquier componente de la fábrica interpretado como sistema ciber físico.

Las tecnologías de la información se integran en la capa ERP y MES de la automatización debido a que en estas capas se maneja y analiza los datos generados por sensores y actuadores, y permite integrar de igual manera con otras áreas de negocio, permitiendo una comunicación entre propietarios, clientes, proveedores y personal que trabaje en la fábrica. Con la utilización de las tecnologías de información se cumple el objetivo de una Industria 4.0 el cual es conseguir que las máquinas permanezcan interconectadas e interactuando entre sí, también el almacenar y analizar los datos e información que es otorgada por sus componentes, permitiendo su monitoreo y manejo en tiempo real y desde cualquier lugar remoto que tenga acceso a internet (Pérez, 2018).

Además, el uso de estas tecnologías facilita el acceso y utilización de una enorme diversidad de servicios remotos y equipos que son vitales para una cuarta revolución industrial y son características implementadas en el proyecto los cuales se menciona a continuación.

Servidor

Un servidor hace referencia a una computadora remota que funciona con una arquitectura cliente servidor es decir, se dedica a responder requerimientos que solicite un cliente, estas respuestas son generadas debido a que el servidor contiene un programa que se está ejecutando de forma perpetua y continua, y está siempre dispuesto a responder peticiones que realice un cliente, estas peticiones son enviadas mediante redes LAN o el internet, en los servidores son capaces de almacenar archivos digitales, aplicaciones, datos y servicios los cuales el servidor deberá responder de manera correspondiente al contenido solicitado por el cliente, los servidores también son capaces de desplegar servicios web las cuales son porciones de códigos contenidas en el servidor y son ejecutadas al realizar una petición o respuesta (Romero, 2015). El servidor del presente proyecto se encuentra alojado en un lugar remoto en él se encuentra todo el backend con la lógica de la fábrica, la base de datos con la información de cada componente de la fábrica y el bróker MQTT que permite la comunicación entre capa física y lógica y el software Ignition actúa de cliente.

- Cliente y servidor pueden estar en máquinas físicas distintas.
- El cliente no necesariamente necesita conocer el programa que contiene el servidor, tan solo interactúa con su interfaz externa.
- El servidor presenta a todos sus clientes las mismas funciones.
- El servidor mantiene su funcionamiento continuo, para no interrumpir los servicios a sus clientes.

Servicios Web

Son funciones que se encuentran albergadas en un servidor en la que diferentes servicios o dispositivos la utilizan, están diseñados para comunicarse sin importar en que lenguaje este escrito el software en el servidor y el lenguaje de donde sea consumido por el cliente. En el caso de los Servicios Web RESTful, son servicios basados en REST y estos

servicios responderán recursos los cuales son una entidad almacenada, solo es necesario que el cliente solicite el recurso mediante un método HTTP sea este get, post, put o delete, un URI y un parámetro dependiendo del método, por ejemplo se desea solicitar un recurso de un servicio web REST, se ingresa la dirección en el navegador, esta realiza un método get, esta petición estará siendo realizada al servidor el cual responderá con el recurso solicitado por el cliente en formato XML o JSON, de igual manera al realizar un método post, se ingresa en formato JSON el recurso que se desee agregar y con la URI correspondiente y este recurso será agregado por el servicio web, el método put permitirá actualizar cualquier recurso y delete borrar el recurso, en conclusión los servicios web son importantes y son fáciles de manejar debido a tan solo se necesita la URL y un método para acceder a los recursos además que tienen alta interoperabilidad, esto permite tener escabilidad sin complicaciones y el lenguaje utilizado es común como es el JSON, además que sus respuestas son códigos de HTML como son 200,201,404 que son códigos muy conocidos por los programadores (Haro et al., 2019).

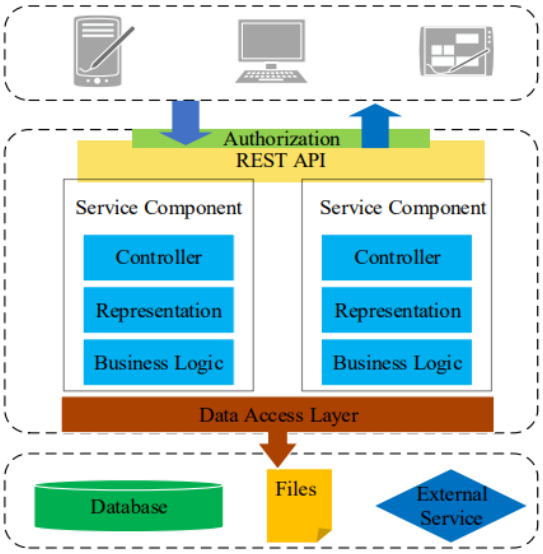
- Tienen interoperabilidad ya que no importa el lenguaje en que este escrito el servicio web ni el lenguaje que utilice el cliente para consumir el servicio.
- Se trabaja con facilidad y desde cualquier lugar ya que casi todos los dispositivos soportan HTTP.
- Alto grado de escabilidad ya según el proyecto vaya aumentando solo se deben aumentar servicios web independientes.
- Los códigos en que responde el estado del servicio son conocidos ya que son los mismos de HTLM.
- El programa que contiene los servicios web puede estar muy alejado de donde serán consumidos estos servicios.

En la Figura 12 se muestra una representación de un cliente que consume un Servicio Web Rest y que está ligado a una base de datos y a otros servicios externos, en el presente

proyecto los Servicios Web son muy importantes debido a que estos realizan el almacenamiento de la información recopilada de los componentes de la fábrica, vistos como componentes web, el despliegue de la información almacenada de cada componente en formato JSON cada uno con una URL y la virtualización de los sensores y actuadores de la fábrica.

Figura 12

Diagrama de Servicio Web Rest



Nota. Figura tomada de (Binfeng et al., 2017).

Bases de Datos

Tecnología la cual permite el almacenamiento de datos puede estar a nivel local o remoto, con datos estructurados o no, en la fábrica inteligente permite el almacenamiento masivo de información generado en un ciclo productivo por los sensores y actuadores los cuales son los encargados de recopilar y enviar esta información a la nube de cómputo, esta información en su mayoría son conjuntos de datos complejos y diversos generados por una gama de instrumentación en el nivel físico que estará disponible en cualquier lugar del mundo y

también estará disponible para cualquier plataforma, estos datos son de gran importancia debido a que permite realizar gráficas de cada componente en tiempo real y con un correcto análisis estadístico con modelos predictivos, solventar problemas existentes o que antes no hubieran sido tratados a tiempo, una base de datos se encarga de guardar datos del pasado y presente de la fábrica para desplegar estos datos en tiempo real y dar al operario una idea de que decisión tomar por ejemplo, ver la gráfica en la que se descalibro un sensor del nivel de llenado y parar la producción y dar atención al mismo, además de mediante análisis de estos datos poder predecir el futuro en la parte operativa de la planta, también influye en la parte administrativa para encontrar patrones hacía los que se inclinan los clientes y tomar decisiones por el personal para aumentar la eficiencia de la planta, permite sustituir y asistir al operario en algunas condiciones para preservar su seguridad física o en tareas poco relevantes que se han venido haciendo de forma manual con un patrón idéntico haciendo uso de los datos recopilados y su debido análisis dar asistencia a un mantenimiento predictivo, dando los beneficios de mayor tiempo de funcionamiento, mayor eficiencia y mayor productividad (Viktor , 2013).

Correo Electrónico

Entre los servidores remotos que el proyecto integra en la capa lógica se tiene el correo electrónico el cual permite el intercambio de mensajes información o archivos digitales, el protocolo utilizado es el SMTP, estos correos en una Industria 4.0 son generados en base a la información del ciclo productivo de manera automática para informar de reportes a clientes, proveedores, gerentes, operarios, jefes de mantenimiento el estado del automatismo, estos reportes llevan información muy importante la cual puede ser, alarmas, orden de producción realizada, mantenimiento, índices de productividad, escasez de materia prima, son enviados mediante una notificación de correo electrónico y se encuentran siendo coordinados con el orquestador con un módulo de Node JS llamado Node Mailler, el cual es configurado mediante el SMTP de Gmail y con los permisos que este requiere para poder enviar mensajes además

tiene la posibilidad de que el remitente pueda enviar varias notificaciones y puedan ser varios destinatarios a la vez de ser necesario, estos correos son parte del servicio web del orquestador el cual mediante sus entradas y salidas realizará el envío automático de los correos electrónicos.

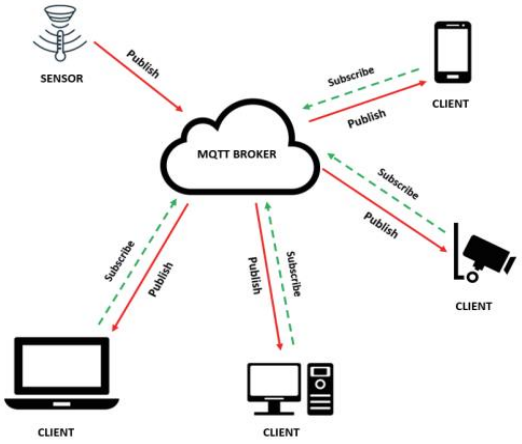
MQTT

Es el protocolo de mensajería asíncrona ligera más utilizado en lo que corresponde a diseños IoT y de comunicaciones M2M o máquina a máquina, su modelo se basa en la publicación y suscripción, su implementación permite ser utilizada con dispositivos de recursos limitados así como de ancho de banda reducido y con latencias relativamente altas, además MQTT funciona sobre el protocolo TCP/IP lo cual es beneficioso para que se puedan comunicar dispositivos completamente diferentes y que utilizan lenguajes de programación diferentes, por lo cual es muy ventajoso su uso para el envío de datos de sensores a servidores remotos, el funcionamiento de MQTT es simple los mensajes o dispositivos a enviar se clasifican según un "tópico" el cual es el tema del mensaje, es decir a quien va dirigido o de quien se desea recibir el mensaje, están conectados en forma de una red estrella en la que el nodo central es un Broker el cual gestionará el tráfico de los mensajes que le lleguen de cada tópico y a manera de un Hub los encaminará a los dispositivos que estén suscritos, utilizar MQTT trae muchos beneficios como son los siguientes (Mahedero, 2020).

- Requiere ancho de banda mínimo por lo que puede ser implementado en redes con latencia alta o problemas de calidad.
- Tiene un impacto alto actualmente por lo que la mayoría de dispositivos IoT están integrando este protocolo.
- Consume poca energía, muy recomendable en dispositivos que usen baterías.
- Existe un intermediario llamado broker el cual es el servidor encargado de gestionar el tráfico de los mensajes.

Figura 13

Transmisión MQTT



Nota. Figura tomada de (Chi, Dang, & Vu, 2022).

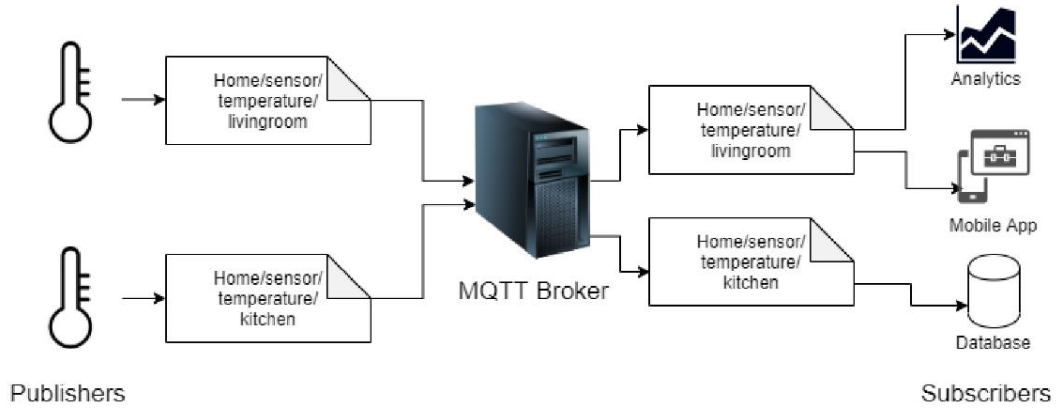
Broker MQTT

Para la utilización del protocolo de comunicación MQTT en el proyecto es fundamental la utilización de un broker propio, el cual es un servidor de mensajes encargado realizar la gestión de la red, de recibir los mensajes enviados y distribuirlos, utilizando el modelo de publicación / suscripción, el canal siempre se mantiene activo debido a que los clientes envían periódicamente un paquete de datos y esperan la confirmación del servidor o broker, la mayor ventaja que proporciona utilizar un broker es el desacoplo entre transmisor (publicador) y receptor (suscriptor) debido a que el broker gestiona el tránsito de los mensajes además el servidor guarda el mensaje y espera a que el cliente se conecte, es muy utilizado para el intercambio de mensajes entre dispositivos de recursos limitados sea por su potencia o por el ancho de banda y por esta razón generalmente utilizado para componentes IoT (Biswajeeban & Attila , 2021). En este trabajo se utilizará el broker el cual es un software llamado Mosquitto

- El broker puede encontrarse alojado de manera local, instalado en un servidor remoto o basado en la nube utilizando los principales proveedores de servicios en la nube como IBM, Microsoft, etc.
- La mayoría de broker son ligeros y no consumen muchos recursos del servidor en el que estén instalados.
- El broker gestiona todo el tráfico de mensajes, y es escalable fácilmente puede pasar de un dispositivo a miles
- Según sea el tópico envía la información correspondiente como un hub.
- Se puede configurar una autentificación y una encriptación SSL/TLS por lo cual hace que el envío de mensajes MQTT sea muy seguro.

Figura 14

Componentes MQTT, Broker



Nota. Gráfico obtenido de (Biswajeeban & Attila , 2021).

Tecnologías Operativas (OT)

Tecnologías que en su mayoría son usadas para la interacción con el mundo físico al contrario con las tecnologías de la información que se dedica al tratamiento de la información, estas tienen la función de detectar cambios en sus dispositivos de nivel 1 o nivel físico es decir sensores y actuadores, las tecnologías operativas son capaces de monitorear y controlar a

dispositivos físicos de uso industrial los cuales son robustos para resistir un ambiente industrial en condiciones ambientales no óptimas y con un uso continuo, suelen estar aislados para precautelar la disponibilidad del proceso y que su producción sea continua, debido a que si algún dispositivo falla, se detiene toda la cadena. Se pueden definir también como una combinación de hardware y software que permite la automatización de una cadena de producción, controlan cada máquina o dispositivo que forme parte del proceso productivo, verificando su funcionamiento normal y si existen condiciones que requieran la atención de algún operario, las tecnologías operativas que fueron utilizadas y algunas simuladas en el presente proyecto están conformadas por lo siguiente.

Controlador Lógico Programable (PLC)

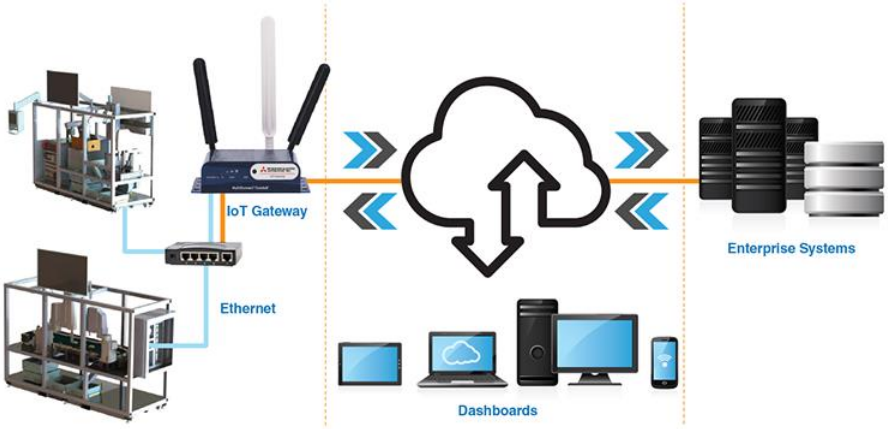
El PLC es el bloque de construcción clave de un proceso de automatización siendo el cerebro de una máquina de una etapa de la cadena de producción, no es más que un computador creado con una placa electrónica y circuitos integrados que permiten la obtención de valores físicos, valores que son entregados por sensores que a su vez el PLC convierte con sus canales ADC a digitales para realizar el control de procesos, encender y apagar actuadores según las condiciones que el proceso demande pero aparte de realizar estas tareas debe ser robusto para un uso industrial, ser capaz de compartir lugar con motores, ruido electromagnético que estos generan, polvo, y condiciones no óptimas que tiene una línea de automatización, los PLC hasta ahora solo controlan los actuadores mediante bucles de control, retroalimentados por sensores y siguiendo un conjunto de instrucciones creadas en un lenguaje de programación llamado Ladder o lenguaje de escalera obedecen estas instrucciones para cualquier sistema de automatización.

Los PLC concentran toda la información de datos de sensores y actuadores que desgraciadamente se queda encerrado en las fábricas, además que estos no tienen interoperabilidad es decir un sistema SCADA de un proveedor solo funciona con un PLC del

mismo proveedor, los PLC tienen conexión para crear redes industriales pero nunca fueron creados para ser integrados al internet de las cosas como tal, es por esto que se hace uso de una pasarela, la cual es fundamental para poder integrar el PLC al IoT pues una pasarela se encarga de romper la brecha entre la OT y la IT, de esta manera las etiquetas del PLC los envían a la pasarela, la pasarela transfiere los datos en un tiempo periódico a internet mediante diferentes protocolos a elegir como puede ser HTTP o MQTT (Kubr et al., 2021).

Figura 15

Pasarela o Gateway para uso industrial



Nota. Figura obtenida de (mitsubishielectric, 2022).

Sensores

Los sensores son dispositivos capaces de medir una magnitud física y proporcionar una señal útil como puede ser el voltaje o la corriente con respecto a la cantidad de la magnitud física medida para que puede ser utilizada por el operador, en los sistemas de automatización los sensores miden variables físicas propias de proceso tales como el nivel, presión, flujo y estos valores son transformados por el PLC con los canales análogos digital en el caso de los sensores análogos, en el caso de los sensores digitales son leídos como un 1 o 0, esta información ya es legible para la lógica de programa del PLC el cual tomará decisiones en torno

a los sensores que tenga conectado, se encuentran en la capa física del proyecto pues intervienen con el proceso físicamente es decir pueden palpar las condiciones del proceso, serán los encargados de transformar las magnitudes físicas en señales eléctricas que después el PLC realizará la adquisición y recopilación de las mismas y finalmente enviará mediante el Gateway al servidor (Kaschel & Pinto, 2020).

Respecto a los sensores del proyecto se realizó una virtualización de los mismos pues son simulados por un servicio web para emular los datos que deberían enviar si fueran implementados. En la Tabla 6 se muestra la lista de los sensores que son utilizados en el caso de estudio para comprobar el funcionamiento de la arquitectura, el caso de estudio es la etapa de envasado de una embotelladora, la lista cuenta con descripción y el tipo de sensor que fue simulado.

Tabla 6

Lista de sensores del caso de estudio

Código	**Descripción**	**Tipo**
sensor00	Botón inicio	Digital
sensor01	Botón automático	Digital
sensor02	Botón Manual	Digital
sensor03	Botón emergencia	Digital
sensor04	Sensor presencia de botellas 1	Digital
sensor05	Fin de carrera extendido de cilindro 1	Digital
sensor06	Fin de carrera retraído de cilindro 1	Digital
sensor07	Fin de carrera extendido de cilindro 2	Digital
sensor08	Fin de carrera retraído de cilindro 2	Digital

Código	Descripción	Tipo
sensor09	Fin de carrera extendido de cilindro 3	Digital
sensor10	Fin de carrera retraído de cilindro 3	Digital
sensor11	Fin de carrera extendido de cilindro 4	Digital
sensor12	Fin de carrera retraído de cilindro 4	Digital
sensor13	Fin de carrera extendido de cilindro 5	Digital
sensor14	Fin de carrera retraído de cilindro 5	Digital
sensor15	Sensor de nivel llenado 1	Análogo
sensor16	Sensor de nivel llenado 2	Análogo
sensor17	Sensor de nivel llenado 3	Análogo
sensor18	Sensor de nivel llenado 4	Análogo

Actuadores

(Jácome, 2019) menciona que un actuador es un dispositivo industrial que transforma la energía en fuerza sobre un elemento mecánico para moverlo, el controlador realiza un impulso y el actuador realiza una acción, la energía que permite este movimiento proviene de 3 tipos: eléctrica, neumática e hidráulica, estos movimientos pueden ser lineales o rotativos, el encargado de realizar el impulso que mueva los actuadores es el cerebro de la automatización y casi siempre lo realiza el PLC, los actuadores nos dan muchos beneficios como son:

- Existe un sin número de actuadores conforme la necesidad de la automatización, además el conjunto de actuadores lineales y rotativos puede servir para realizar movimientos complejos.
- Los actuadores permiten controlar el paso de la automatización algunos permiten su regulación en velocidad o fuerza.

- Actuadores neumáticos utilizan aire comprimido mientras que los hidráulicos utilizan aceite o líquido especial para controlar el movimiento.
- Los actuadores son comandados por el PLC el cual envía los impulsos y controla cuándo cómo y dónde deben ejercer movimiento.

Tabla 7

Lista de actuadores del caso de estudio

Código	**Descripción**	**Tipo**
actuador00	Luz Paro	Luz Piloto
actuador01	Luz Funcionando	Luz Piloto
actuador02	Luz Emergencia	Luz Piloto
actuador03	Banda Transportadora	Eléctrico
actuador04	Cilindro de paro de botellas con contador	Neumático
actuador05	Cilindro que introduce las válvulas de llenado	Neumático
actuador06	Cilindro de paro de botellas llenadas	Neumático
actuador07	Cilindro para tapado a presión	Neumático
actuador08	Cilindro para paro de botellas tapado	Neumático
actuador09	Válvula de llenado 1	Eléctrico
actuador10	Válvula de llenado 2	Eléctrico
actuador11	Válvula de llenado 3	Eléctrico
actuador12	Válvula de llenado 4	Eléctrico

La principal desventaja de las tecnologías operativas es que cada máquina está aislada y son consideradas como islas es decir están incomunicadas unas a las otras. Para que se logren implementar los beneficios de una industria 4.0 es necesario una convergencia entre IT y OT y es precisamente donde los componentes de red de las Tecnologías Operativas, los sistemas

de control, SCADA y la red industrial deben ser integrados con los componentes de las Tecnologías de la Información como servidores, procesadores, almacenamiento en base de datos y demás sistemas implicados para que se permita una industria 4.0.

Arquitectura Orientada a Servicios (SOA)

SOA otorga una arquitectura de software en la que lo primordial es la utilización de servicios web independientes que siendo integrados permiten que trabajen en colaboración para construir aplicaciones de software en distintos sistemas permitiendo la interoperabilidad. SOA gracias a la utilización de una capa de servicios permite a los sistemas y servicios interactuar a través de la red y reutilizar elementos, esto otorga muchos beneficios como el hecho de no trabajar de manera directa con ellos puesto a que si dos servicios desean comunicarse no es necesario conocer la lógica con la que funciona el otro servicio y utilizarán la capa de servicios como intermediario ya que esta capa cuenta con las especificaciones de cómo funciona cada servicio permitiendo acceder de forma remota o actualizar los servicios de manera independiente todo esto para dar soporte a los requisitos de un negocio.

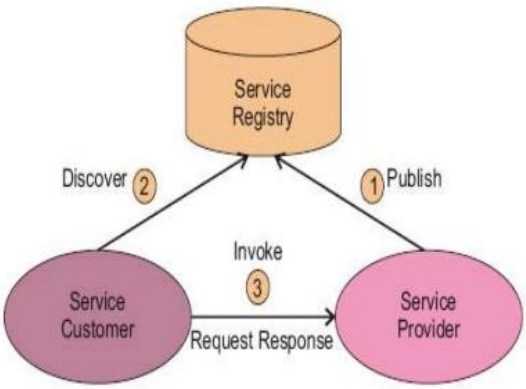
(Sudhanshu & Mukherjee, 2019) menciona que SOA presenta una solución en la que lo primordial es el uso de servicios dado a la interoperabilidad que estos ofrecen como es que puedan estar escritos en diferentes lenguajes de programación y aun así entenderse, SOA estará formada por una combinación de APIs, esto aportará algunos beneficios como se dicta a continuación.

- Esta arquitectura permite que los Servicios Web sean independientes entre ellos y puedan intercambiar datos, aún si han sido escritos en lenguajes de programación diferentes.
- Permite tener una gran escabilidad debido a que integra servicios web implementados por separado y permite crear aplicaciones trabajando en conjunto para distintos sistemas.

- Tienen un mantenimiento sencillo debido a que son independientes y si fuera necesario actualizar alguno se lo realiza por separado.
- Desarrollo de aplicaciones más rápido debido a que el programador no tiene que iniciar de cero siempre pues se puede reutilizar servicios existentes
- Otorga mayor confiabilidad ya que es más fácil depurar errores en servicios pequeños que en un código monolítico.
- El cliente sabe cómo utilizar el servicio, que información se debe enviar y que responde, pero no que tiene dentro ni con que lógica funciona.
- SOA da una alta flexibilidad, pero no es óptima en tiempos, debido a que existen servicios que funcionan dependiendo de otros servicios llamado servicios compuestos y al invocar a estos, existe un tiempo de latencia ya que algunos servicios se encuentran muy alejados.

Figura 16

Colaboración de servicios en un entorno SOA



Nota. Figura Obtenida de (Saleem, 2017).

Como se puede observar en la Figura 10 SOA es una arquitectura la cual permite la integración de servicios estando cada uno independiente del otro y manteniéndose separados, permite la

comunicación entre estos y que en forma conjunta se pueda desarrollar aplicaciones complejas y que se encuentren en distintos sistemas.

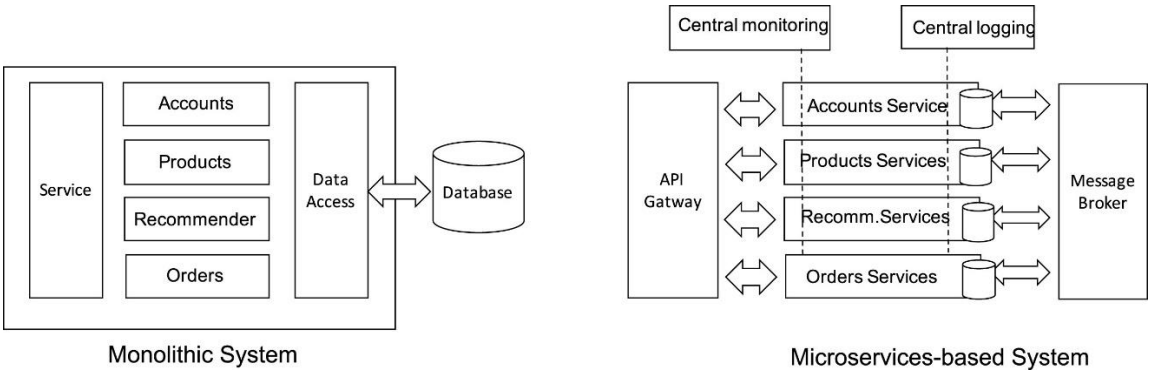
Microservicios

Los microservicios son pequeños servicios que trabajan en conjunto con un fin en común. Los microservicios buscan descomponer una aplicación monolítica por un conjunto de servicios independientes, como se observa en la Figura 17 los microservicios generan muchos beneficios ya que al tener una aplicación monolítica un cambio por pequeño que sea requiere una reconstrucción total del código y que este permanezca apagado hasta realizar los cambios en el monolítico, además que una aplicación monolítica es más difícil de mantener y difícil de escalar, mientras los microservicios si se cae un servicio no afecta a la aplicación en los demás servicios y hasta pueden ser creados en diferentes plataformas tecnológicas, lenguajes de programación, pueden escalar independientemente, son fáciles de mantener y se dicen que son tolerantes a fallas ya que una falla en un servicio no interrumpirá el sistema en general, cada uno se ejecuta por separado y podrían estar siendo reutilizados (Jacopo et al., 2018).

(Mazzara et al., 2020) menciona que cada microservicio puede ser considerado como una pila monolítica independiente el cual requiere un entorno completo, por ejemplo, una propia base de datos para lo cual se realiza una coordinación para que estos trabajen juntos para el funcionamiento de la aplicación a la cual fueron creados, esto se llama orquestación. El orquestador es el cerebro del proyecto presente pues se realiza una orquestación de los microservicios de sensores, actuadores y servidores remotos como el correo electrónico, esta coordinación de servicios web permitirá que el orquestador realice el control industrial del caso de estudio propuesto, integrando las tecnologías operativas con las tecnologías de la información.

Figura 17

Comparación de sistema monolítico y sistema basado en microservicios



Nota. Figura tomada de (Mazzara et al., 2020).

Los microservicios se comunican en mayor frecuencia utilizando REST con el protocolo HTTP con los métodos antes mencionados en servicios web como get, post, delete, put, los microservicios manejan entonces todas las peticiones HTTP y ejecutará la lógica del dominio, recupera, guarda y actualiza información de la base de datos, y en caso del proyecto presente estos guardan y despliegan la información de la base de datos la cual ha sido guardada producto de la recopilación de la virtualización de los sensores y los actuadores. Los microservicios tienen grandes beneficios como se menciona a continuación.

- Descompone una aplicación monolítica en servicios por lo cual se puede desplegar, modificar sin interrumpir los demás servicios.
- Mantiene un sistema descentralizado, cada módulo tiene su propio entorno evitando que se pueda sobrecargar de solicitudes.
- Fácil integración y permite el trabajo en equipo pues se puede repartir el trabajo de forma más fácil.
- Su coordinación permite que estos interactúen entre sí, muy importante para un proyecto como un diseño de IoT en el que se busca que todo este interconectado.

Sistemas Ciberfísicos (CPS)

(Napoleone, Macchi, & Pozzetti, 2020) menciona que los sistemas ciberfísicos se consideran dispositivos de nueva generación, un sistema ciberfísico es todo dispositivo el cual integra en sus capacidades la comunicación, el software y el almacenamiento, los CPS están interconectados, interactúan entre sí, pueden integrarse a servidores remotos para el almacenamiento información, esta información es generada por los sensores y actuadores que el sistema dispone, es decir están en contacto con el medio físico con transductores y convierten las magnitudes físicas del entorno en una señal digital para que sea enviada a redes de internet y que esté disponible en la red, esta característica de los sistemas ciberfísicos permite tomar decisiones y mediante los actuadores del CPS influir físicamente en su entorno y al poder tener toda la información en la red permite una monitorización del proceso en tiempo real, una característica importante que deben tener los sistemas ciberfísicos es que a diferencia de los sistemas embebidos que realizan una tarea y su lógica en muchas ocasiones es imposible de cambiar, los CPS deben ser flexibles y está lógica pueda ser fácilmente modificada en la nube y que también deben ser altamente escalables e integrarse con tecnologías emergentes.

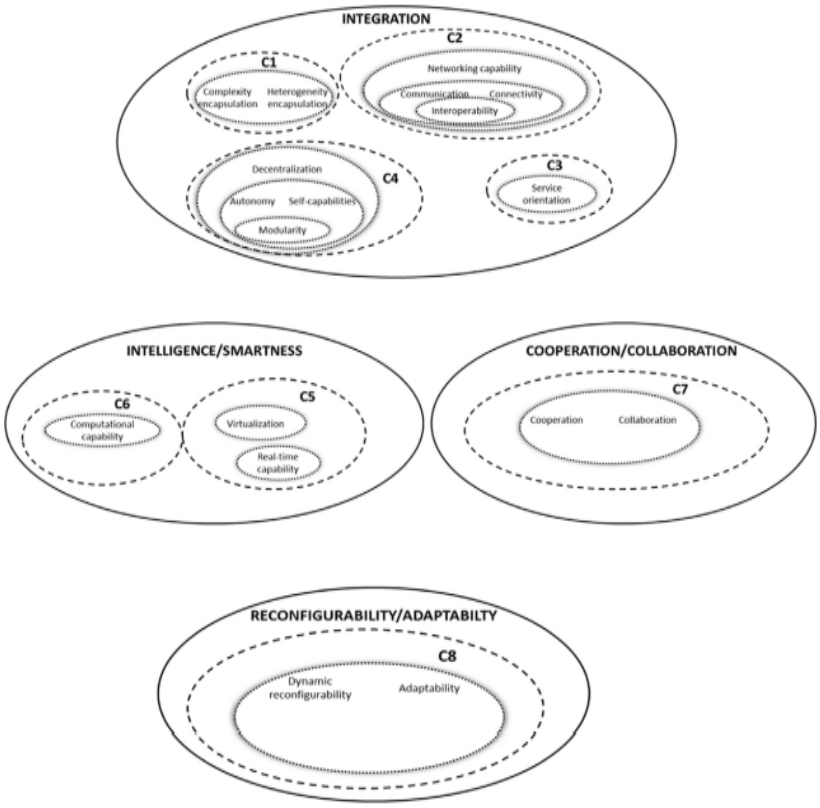
Partiendo de un sistema ciberfísico tenemos los sistemas de producción ciberfísicos (CPPS) estos son sistemas los cuales son considerados como los componentes básicos de una fábrica inteligente, permiten integrarse a una línea de producción con el fin de brindar mayor flexibilidad, eficiencia a un proceso productivo, son capaces de tomar decisiones por cuenta propia y también de obedecer las acciones enviadas por la interfaz humano-máquina, también deben dar asistencia al operario y tener capacidad de interpretar si este corre algún tipo de riesgo o emergencia, además se dice que la integración de estos sistemas se logra mediante el uso de tecnologías como sistemas multi agente (MAS), arquitectura orientada a servicios (SOA) y el internet de las cosas (Hozdic & Butala, 2020).

Los sistemas de producción ciberfísicos rompen la pirámide de automatización tradicional en los cuales ahora el PLC que se encuentra en el nivel de control y el campo es ahora utilizado solo para tareas que requieran un lazo de control crítico o de emergencia y todos los demás componentes deben ser sistemas ciberfísicos y deben estar más descentralizados puesto a que deben ser independientes entre ellos, pero interactuar entre sí.

Según (Napoleone, Macchi, & Pozzetti, 2020) menciona que las características que deben tener los sistemas de producción ciberfísicos como se muestra en la figura 18 para fábricas inteligentes son: la integración, la inteligencia, colaboración entre sistemas ciberfísicos y tener la facilidad de ser reconfigurables, que tengan comunicación hacia el exterior pues deben funcionar con tecnologías como la IoT o arquitecturas basadas en servicios, además deberán tener un sistema computacional embebido que es el encargado de convertir las señales de los sensores en digitales, de manejar los actuadores con la potencia requerida, recopilar esta información y enviarla a la red de manera periódica y que su coordinador es decir el que tiene la lógica del dispositivo esté albergado desde un servidor para que su programa sea flexible y permita cambios desde la nube, realizar procesamiento de datos y análisis de la información recopilada y guardarla en la base de datos, en conclusión los sistemas de producción ciberfísicos son la célula de una fábrica inteligente tienen la capacidad de relacionarse con un entorno industrial descentralizado en el que cada componente ciberfísico es independiente y a la vez trabaje en conjunto para mejorar la producción, flexibilidad y eficacia de una industria es por ello que se conceptualiza cada componente del caso de estudio de la Smart Factory como un componente ciberfísico y se toma de la arquitectura de los microservicios para la realización del presente trabajo ya que esta arquitectura tiene características que se busca que realice los componentes ciberfísicos en una fábrica inteligente.

Figura 18

Características de un sistema ciberfísico para una fábrica inteligente



Nota. Figura tomada de (Napoleone, Macchi, & Pozzetti, 2020).

Internet de las Cosas (IoT)

Es un sistema en el cual existe un conjunto de dispositivos interconectados a través de una red, son capaces de interactuar, autoorganizarse y actuar ante situaciones y cambios en su entorno. IoT describe una red de interconexión a la que cualquier cosa puede conectarse al referirse a cosa este es un dispositivo el cual puede ser tan complejo como un robot con decenas de sensores y actuadores, o tan simple como una refrigeradora que envía sus datos de temperatura y recibe la hora de la nube, esta tendencia permite que se generen una enorme cantidad de datos que antes no eran analizados debido a que los datos que generaban las

cosas casi siempre se quedaban dentro del dispositivo y microcontrolador tenía baja capacidad de almacenamiento.

(Somayya et al., 2015) menciona que los requisitos previos para que exista Internet de las cosas son: Necesidades de desplegar datos en tiempo real, Crecimiento exponencial de demanda, Disponibilidad de aplicaciones, Protección de datos, Consumo eficiente de la energía, Acceso a un sistema en la nube, mientras tanto en hardware se necesita: Sensores y actuadores, Middleware, almacenamiento en base de datos y computación para análisis de datos y una Presentación o capa de aplicación para mediante la cual se pueda mostrar los resultados y sea fácil de interpretar los datos recopilados.

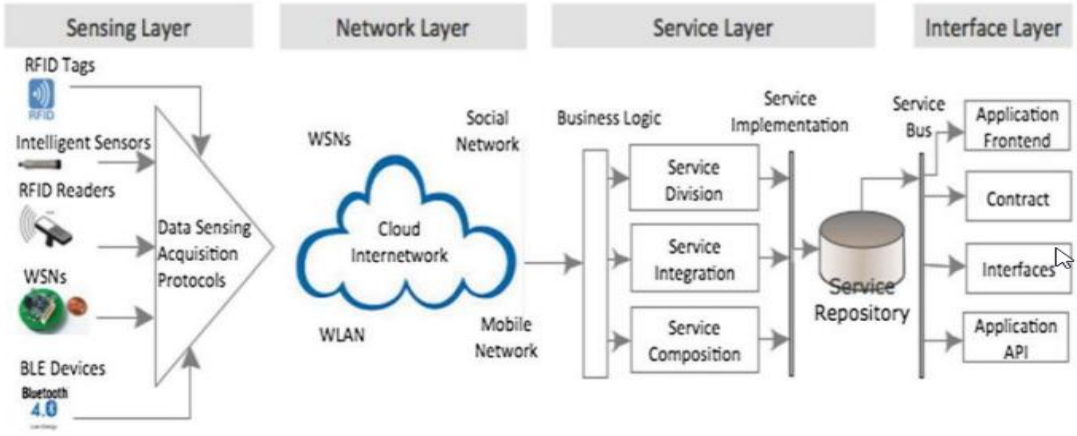
Un requisito que se necesita en el internet de las cosas es la interconexión, y por esto es crucial la arquitectura a utilizarse, dado que la arquitectura elegida debe garantizar este requisito además de poder ser escalable e interoperable entre dispositivos que no utilicen necesariamente el mismo protocolo de comunicación, esta arquitectura se desea que sea adaptable y que tenga una comunicación inequívoca de eventos. Es por ello que (Gokhale, Bhat, & Bhat, 2018) menciona que la utilización de la arquitectura orientada a servicios en el internet de las cosas es ampliamente utilizada como arquitectura principal, la misma se muestra en la Figura 19 y está formada de las siguientes capas:

- Capa de detección dispone físicamente de los objetos y detecta cambios físicos de estados en sus sensores.
- Capa de red dispone de una arquitectura para admitir las conexiones de las cosas es decir el envió o recepción de datos.
- Capa de servicio en los cuales se administran los servicios requeridos para almacenamiento, despliegue y orquestación.

- Capa de interfaz permite interactuar entre el usuario y la aplicación, tener el control de los actuadores en el entorno que se encuentren y el monitoreo en tiempo real de los datos.

Figura 19

Capas de la arquitectura de IoT



Nota. Figura obtenida de (Gokhale, Bhat, & Bhat, 2018).

En cuanto a los protocolos de comunicación que se utilizan en IoT, los más habituales son MQTT, CoAP y HTTP, debido a que son altamente flexibles, en el presente proyecto se utiliza la arquitectura orientada a servicios como arquitectura para implementar el internet de las cosas, el protocolo comunicación que tendrán los dispositivos será mediante el protocolo MQTT, se utilizan microservicios web RESTful los cuales son los encargados de desplegar y almacenar la información que será simulada por servicios web de los sensores y los actuadores, estos microservicios son coordinados mediante un orquestador el cual realizará el control industrial del proceso industrial en la nube, finalmente el proceso puede ser monitoreado mediante Ignition, un software utilizado para crear sistemas SCADA el cual está suscrito a los tópicos de los sensores y actuadores presentes en el proyecto y podrá desplegar la información proveniente en formato JSON en etiquetadas, permite el monitoreo de los

sensores y actuadores en tiempo real mediante la información almacenada en la base de datos en forma de gráficas históricas.

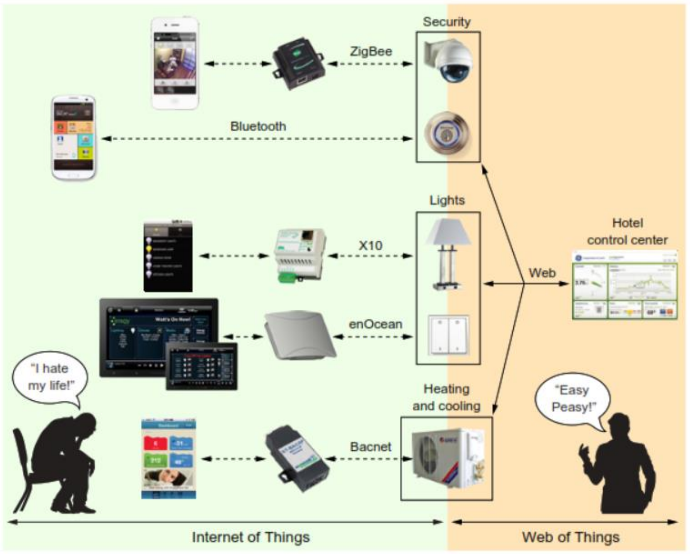
Web de las Cosas (WoT)

La Web of Things es una especialización del Internet de las Cosas, su objetivo principal es definir como acceder, interconectar e integrar dispositivos heterogéneos utilizando tecnologías web estándar, ya que hoy en día coexisten cientos de protocolos y estándares aplicados en la IoT que al momento de integrarse resulta imposible como se muestra en la Figura 20 en la IoT sus tecnologías están fragmentadas en todas sus capas sea física, de red, de transporte y de aplicación, es por ello que la WoT crea un acceso por encima de las capas de la arquitectura de la IoT en la que se base en reutilizar la misma web para integrar dispositivos independientemente de sus protocolos de comunicación, capas haciendo uso de protocolos web comunes como el HTTP y los mensajes estandarizados como en formato JSON, el cual es el formato de intercambio de datos más exitoso en internet lo que otorgaría muchos beneficios como la interoperabilidad y logra abstraer la complejidad de cómo se conecta y comunica el dispositivo y que el programador se centre únicamente en la lógica y la aplicación de los servicios otorgados (Martins, Mazayev, & Correia, 2017).

(Guinard & Trifa, 2016) mencionan que la WoT permite a los desarrolladores y aplicaciones el intercambio de datos con cualquier objeto físico mediante solicitudes HTTP sin importar como esté conectado este dispositivo, el éxito de ocupar la WoT radica en las décadas de desarrollo que tiene la World Wide Web y tiene como objetivo abarcar todos los dispositivos de IoT como sea posible convirtiéndolos en componentes web que puedan ser reconocibles, accesibles e interoperables. Existen tres requisitos para que WoT sea funcional, los cuales son los siguientes.

Figura 20

Integración de cosas en la WoT



Nota. Figura tomada de (Guinard & Trifa, 2016).

- Flexibilidad: se pueda utilizar la WoT en tantos dispositivos como sea posible independientemente de las tecnologías que el dispositivo tenga implementado.
- Compatibilidad: arquitecturas las cuales estén orientadas a puentes entre la IoT y la WoT que generen interoperabilidad con los estándares web actuales.
- Seguridad y privacidad: fuerte énfasis en la seguridad ya que la WoT controla objetos ciberfísicos que interactúan con el mundo físico de manera directa y de ser manipulados por terceros podría tener riesgos enormes

La WoT en la industria permite conectar las máquinas a internet obteniendo los grandes beneficios que resulta poder almacenar información importante de los dispositivos de campo que posean los procesos industriales, además permitiría ver cada componente de la fábrica como un componente web, permite integrar dispositivos que tengan diferentes protocolos de comunicación, y abstraerlos como un componente que puede ofrecer servicios permitiendo utilizar y reutilizar en combinación con otras máquinas resultando en una producción flexible,

además permite que los dispositivos integrados en la web interactúen con cualquier otro servicio web que utilice API web, debido a que estos dispositivos están integrados con los mismos estándares y técnicas que los sitios web tradiciones y en particular arquitecturas RESTful.

Smart Factory

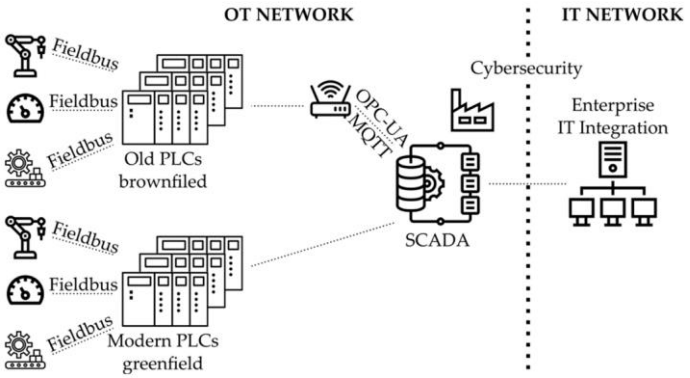
Una Smart Factory es una fábrica la cual integra un conjunto de prácticas de fabricación que utilizan las tecnologías digitales, interconectividad, recopilación de datos, ciberseguridad con máquinas físicas y procesos industriales conectados para gobernar las operaciones de fabricación buscando mejorar la eficacia, eficiencia y flexibilidad a una industria (Osterrieder, Budde, & Friedli, 2020).

Las soluciones para implementar una Smart Factory son el uso de las máquinas, controladores y todo dispositivo de la fábrica sea visto como sistemas de producción ciberfísicos (CPPS), los cuales utilizando su sistema embebido, sensores y actuadores permiten enviar la información a la nube, interactuar entre sí, monitorear y controlar los cambios de producción, los CPS formarán parte de los dispositivos conjuntos del Internet de las Cosas esto mediante la integración de las tecnologías operativas como son los controladores, robots, sensores y todo actuador, combinado con las tecnologías de la información, como se muestra en la Figura 21 los autores muestran cómo se podría llevar a cabo la integración de las dos tecnologías recopilando los datos de PLC’s antiguos y nuevos que envíen su información mediante protocolos de comunicación industrial como Fieldbus o MQTT hacia el sistema SCADA y el mismo envíe toda la información a la nube.

Los componentes en la nube son vistos como una abstracción de una entidad física y pueden ser vistos como servicios web que envían y reciben datos, su coordinación permitirá tener la lógica de control en la nube y estar expuesta para poder ser integrada con otros servicios web y servidores remotos como el Correo electrónico, Amazon, Google, etc.

Figura 21

Convergencia OT y IT en la Industria 4.0



Nota. Figura obtenida de (Cortés, Landeta, & Chacón)

(Mabkhot et al., 2018) menciona que los principios que debe tener una fábrica inteligente son los siguientes.

- Modularidad. Permite a los componentes del sistema estar separados y poder combinarse rápidamente, esta modularidad permite abastecer los requisitos cambiantes del cliente.
- Interoperabilidad. Permite compartir información dentro de los componentes de la fábrica sin importar su tecnología.
- Descentralización. Los componentes del sistema tomarán decisiones sin estar subordinados por un controlador y podrán interpretar que hacer según como se desarrolle el entorno con los demás CPPS.
- Virtualización. Permite crear un entorno de monitoreo artificial de la fábrica en la que se pueda observar los cambios producidos y el estado de cada componente de la fábrica.
- Orientación al servicio. La industria debe adaptarse a las necesidades del producto que el cliente desea, sea en cantidad o en diferentes productos.

- Capacidad de almacenamiento masivo. La industria deberá ser capaz de recopilar y almacenar todos los datos de sus componentes que serán importantes porque mediante su análisis se podrá efectuar la predicción y desenvolvimiento del proceso productivo.

Herramientas de desarrollo

Node.js

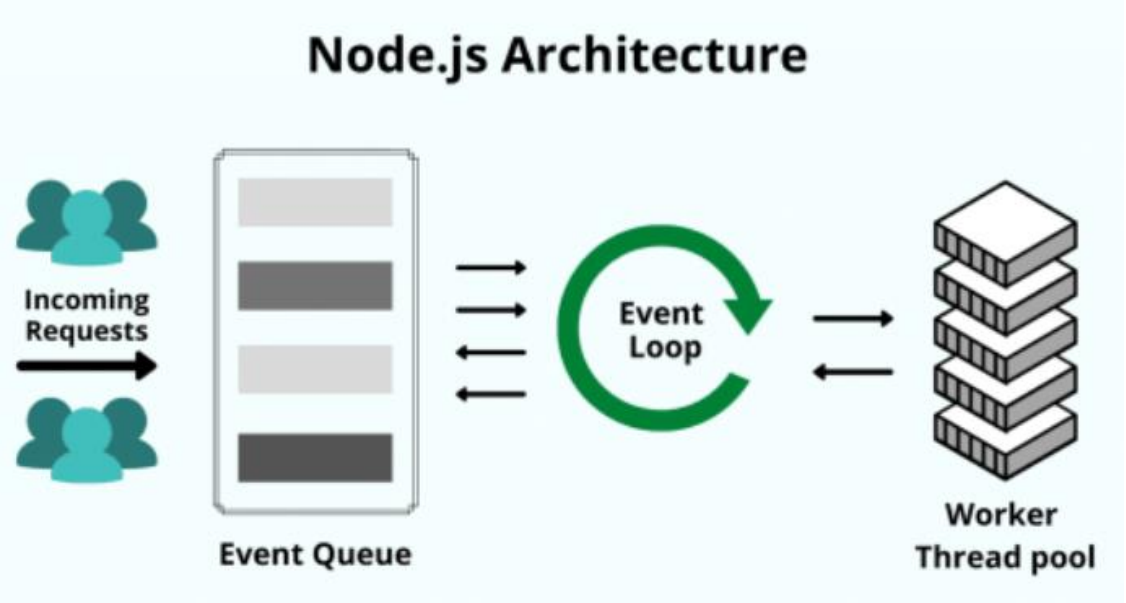
(Chaniotis, Kyriakou , & Tselikas , 2015) menciona que Node.js es un entorno de ejecución controlado por eventos asíncronos con el lenguaje de programación JavaScript creado para crear aplicaciones altamente escalables, Node JS utiliza un modelo solicitud respuesta en base a hilos, es decir cada solicitud de un cliente es procesada y se asigna uno de los múltiples hilos para procesar esta llamada que a veces pueden ser concurrentes, es por esto que es muy utilizado en aplicaciones del internet de las cosas debido a que cada cosa o dispositivo envía pequeños datos al servidor de Node.js y este tiene que ser capaz de gestionar todas las solicitudes que a veces son simultáneas, las características de este entorno son las siguientes.

- Fácil de empezar debido que es altamente ocupada por los programadores por lo que tutoriales y contenido es fácil de encontrar.
- Escalable. Es capaz de manejar muchísimas conexiones simultaneas con alto rendimiento.
- Velocidad. Gracias a su procesamiento multihilos el tiempo de petición respuesta es relativamente bajo perteneciente a los milisegundos.
- Paquetes. Node JS dispone de miles de paquetes que facilitan el trabajo en toda área y esto sigue incrementando.
- Multiplataforma. Se pueda usar en todos los sistemas operativos.

- Mantenimiento. Es fácil dar mantenimiento al código debido al uso de JavaScript en back-end y front-end.

Figura 22

Arquitectura de Node.js



Nota. Figura tomada de (Kinsta, 2021).

Ignition

Es una plataforma de Inductive Automation es un software para sistemas SCADA, permite creación de HMI interactivas en las cuales se puede integrar dispositivos, obtener gráficos históricos y dispone de conectividad a redes industriales como OPC, Modbus TCP y MQTT. Ignition permite la integración de algunas marcas de PLC lo que permite unificar en un solo sistema SCADA diferentes marcas de PLC, además se caracteriza por su rápido desarrollo en los cuales este software se encarga de obtener todas las etiquetas de los PLC que hayan sido conectados y mediante su plataforma modular podemos ir integrando módulos para el tipo de comunicación o requerimientos que se vaya necesitando.

Ignition permite la conexión a protocolos del Internet de las Cosas como IoT, mediante el protocolo de comunicación MQTT es fácil integrar a Ignition y permite el almacenamiento y extracción de base de datos de aplicaciones industriales. En Ignition basta con suscribirse al tópico y si los datos son enviados en JSON Ignition detectará esta Thing Description como etiquetas que pueden ser arrastradas al interfaz humano máquina y mediante los componentes visuales fácilmente mostrados en gráficas, actuadores virtuales o sensores. En el presente proyecto Ignition es el cliente pues está subscrito a los actuadores y sensores, en su interfaz se despliegan los gráficos históricos, la Thing Description de los sensores actuadores, alarmas y mensajes que envía el orquestador que está albergado en el servidor.

Capítulo 3: Diseño

Requisitos de diseño

Como se mencionó en el capítulo 1 la arquitectura de software va a constar de tres capas: Física, Lógica y de Aplicación, para el diseño de cada capa se han fijado requisitos los cuales son esenciales para lograr los objetivos del proyecto y fijar restricciones, estos requisitos son de dos tipos funcionales y no funcionales, a continuación, se plantea entonces en la siguiente Tabla los requisitos de diseño para cada capa que el proyecto pretende integrar.

Requisitos funcionales

Tabla 8

Requisitos de diseño funcionales

Requisitos	Descripción
Requerimientos que debe cumplir la arquitectura en la Capa Física.	• No se implementan físicamente, son simulados. • Sensores y actuadores son simulados por servicios web de emulación • Los sensores simulados deben ser de tipo analógico y digital. • Formato de intercambio de datos en JSON • El protocolo de transferencia es HTTP. • Deben tener nombres genéricos para aplicación en cualquier industria. • Los atributos de la Thing Description son insertados desde la interfaz y pueden ser modificados únicamente en la interfaz. • Existe un microservicio REST por cada sensor y actuador que almacena y despliega los datos.

Requisitos	Descripción
	• Existe una tabla en la base de datos para cada componente de la fábrica. • Mediante cualquier navegador se puede obtener la base de datos completa o el último valor de cualquier sensor y actuador. • Una vez en la nube los sensores y actuadores son vistos como componentes web. • Los únicos sensores no simulados son los botones, su valor depende de su activación en la interfaz.
Requerimientos que debe cumplir la arquitectura en la Capa Lógica.	• La capa Lógica debe ser escrita en el lenguaje de programación JavaScript. • El intercambio de datos y publicación del orquestador debe ser JSON. • Todos los componentes de la fábrica deben comunicarse a la capa lógica por medio de solicitudes HTTP. • La producción debe ser flexible en cuanto al producto final entregado en el caso de estudio. • El control industrial del proceso pasa de ejecutarse localmente a ejecutarse en la nube. • El control de la fábrica debe ser realizado por una orquestación de microservicios. • El automatismo de la fábrica inteligente debe contar con al menos tres modos: automático, manual y de emergencia.

Requisitos	Descripción
	• El modo automático funciona únicamente ingresando una orden de producción y un cliente. • Se puede entrar en modo manual en cualquier punto que se presione el botón sin importar en qué etapa se encuentre el sistema. • Cuando se ingrese a modo manual únicamente se desactivan componentes que sean necesarios. • El modo emergencia, detiene la producción y desactiva todos los componentes de la fábrica. • La lógica programada debe contar con alarmas, luces piloto y mensajes que indiquen información en la que se encuentre actualmente el proceso. • Los servicios de simulación deben ser escritos en la capa lógica, en el archivo de orquestación y los componentes deben simular todas las condiciones que el proceso requiera. • Debe existe un URL por cada componente de la fábrica y cada uno debe tener un microservicio independiente. • Se debe contar con envíos de correos a gerente en caso de alarmas, producción terminada o al proveedor cuando se termine la materia prima.

Requisitos	Descripción
	• El programa debe dividirse en tres archivos: sensores, actuadores y orquestación, más otro archivo que realice conexiones como a la base de datos o a servidores remotos como el correo electrónico. • Esta capa corresponde al back-end del proyecto por lo cual debe estar desplegada en un servidor remoto para afirmar que el cliente se puede encontrar a cualquier distancia.
Requerimientos que debe cumplir la arquitectura en la Capa de Aplicación.	• La interfaz debe contar con monitoreo de todos los componentes en tiempo real. • Se deben ingresar cliente y orden de producción. • Se puede desplegar la Thing Description de cada componente y podrá ser editada desde un formulario. • La interfaz es únicamente es un cliente no almacena información ni realiza funciones de orquestación que deben realizarse en la capa lógica. • Debe ser fácil e intuitiva de usar en lo posible mostrar instrucciones y mensajes de como operarla. • Puede desplegar alertas en condiciones que así lo indique el orquestador. • Muestra el proceso completo visualmente para solución y optimización de fallas que se fueran produciendo en torno van desarrollándose las etapas.

Requisitos	Descripción
	• Cada componente analógico debe mostrar gráficos históricos de su progreso a nivel del tiempo. • Debe contar con luces piloto que demuestren el estado en que se encuentra el automatismo. • Muestra un seguimiento de la producción flexible como lotes listos o en proceso. • Dispone de botones requeridos para levantar el funcionamiento del automatismo, entrar a modos implementados y conectarse al servidor. • La interfaz puede ser vista en cualquier cliente si se instala los programas requeridos de esta capa.

Requisitos no funcionales

En la tabla 9 se presentan requisitos no funcionales en los cuales intervienen propiedades como la usabilidad, escalabilidad, seguridad, mantenibilidad y cualquier restricción que pueda tener el sistema.

Tabla 9

Requisitos No Funcionales

Requisitos	Descripción
Requerimientos que debe cumplir la arquitectura en la Capa Física.	• Todas las solicitudes ingresadas a la capa física deben ser mediante el protocolo HTTP sin excepción. • La base de datos debe tener autentificación para acceder a las tablas y procedimientos de

Requisitos	Descripción
	los componentes de la capa física del proyecto. • Es eficiente en el uso del ancho de banda puesto que solo se publican los mensajes MQTT cuando exista un cambio en una variable o atributo de algún componente. • Los datos simulados deben ser precisos por lo cual las simulaciones deben ser paralelas y se debe tener valores análogos. • Los componentes neumáticos deben contar con sensores fin de carrera como se implementa en la realidad. • Disponibilidad alta en la que la producción no se detenga y se puedan cambiar los atributos de cualquier componente en cualquier momento.
Requerimientos que debe cumplir la arquitectura en la Capa Lógica.	• El mantenimiento de la aplicación es fácil debido a la arquitectura de microservicios usada la cual permite realizar modificaciones sin interrumpir todo el sistema. • Con la utilización de Node.js y su procedimiento de hilos, la capa lógica puede recibir de forma simultánea solicitudes de almacenamiento y despliegue de datos. • La capa lógica debe tener una alta disponibilidad ya que de ella depende el funcionamiento de la fábrica. • Los puertos del servidor en el que se encuentre la capa lógica deben ser accesibles para poder desplegar los datos de los

Requisitos	Descripción
	componentes desde cualquier parte del mundo. • La base de datos se encuentra en el servidor que se encuentre la capa lógica. • Se puede tener una ligera latencia de red es permitida debido al uso de un protocolo como el MQTT. • La seguridad de la capa lógica depende del protocolo de comunicación HTTP por lo cual es altamente seguro. • La capa lógica es eficaz al recibir actualizaciones de cambios de sensores y responder rápidamente en los actuadores. • Se debe presentar documentación con la lógica en diagramas de flujo para que pueda ser modificada por otra persona que quede a cargo de la fábrica si fuera necesario.
Requerimientos que debe cumplir la arquitectura en la Capa de Aplicación.	• Debe tener autentificación para poder manipular y monitorear el proceso. • Respetar colores de norma High-Performance para mejorar la usabilidad. • El sistema deberá barrer cada 5 segundos si no encuentra cambios en los componentes. • El sistema es escalable pues se puede integrar los componentes necesarios para cualquier caso de fábrica. • Los datos deben ser desplegados con periodo menor a 50 ms. • El sistema debe proporcionar mensajes de error y asistencia orientados al usuario final.

Requisitos	Descripción
	• El sistema deberá desplegar un menú popup si se presiona sobre cualquier componente del proceso en el que se mostrarán gráficos históricos y sensores que este lindados al componente.

Diseño de la Arquitectura

El Diseño de la arquitectura se va a dividir de igual manera que los requisitos, en tres capas: física, lógica y de aplicación, como se ha visto en los requisitos se trata de un sistema Restfull por lo cual el protocolo de comunicación utilizado será HTTP y el formato de intercambio de datos será JSON en todas las capas, en la capa física encontraremos los sensores y actuadores y se diseñará su comunicación según el protocolo MQTT, en la capa lógica se encuentra todo el back end del proyecto en el cual se encuentran los microservicios que almacenan la información y despliegan esta información mediante solicitudes HTTP y también se encontrará el orquestador, en la capa de aplicación es necesario crear una interfaz y que se pueda monitorear en tiempo real los componentes de la fábrica, tomando en cuenta los elementos en común de todas las capas se puede decir que toda la arquitectura debe tener estas características:

- Sistema RESTfull, es decir comunicarse mediante HTTP.
- Formato de intercambio de mensajes tipo JSON en todo el sistema.
- Sistema altamente escalable con la posibilidad de agregar mas componentes ciberfísicos o integrar servicios web externos.

Capa Física

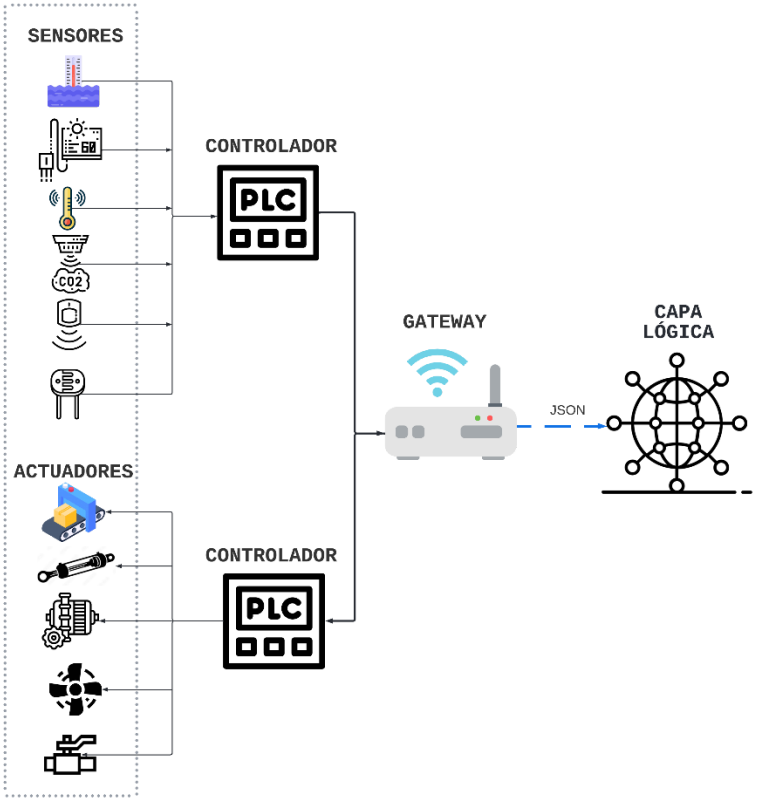
En la capa física de la arquitectura genérica propuesta se encuentran los sensores, actuadores, controladores lógicos programables (PLC) y la pasarela o Gateway como se puede

observar en la Figura 23, los sensores y actuadores van a estar colocados en el entorno físico de la fábrica y el PLC va a adquirir mediante sus canales ADC las señales de los sensores digitales o análogos; estos sensores tendrán cada uno una Thing Description la cual permitirá representar a los sensores como componentes ciberfísicos de la Web Of Things. El controlador o PLC enviará mediante cualquier protocolo de comunicación industrial al Gateway la información en formato JSON de la Thing Description de los sensores, y lo hará solo y cada vez que detecte un cambio en el sensor, es decir en alguno de sus atributos de la Thing Description o en la variable física que ese sensor este involucrada. Esto es realizado para aprovechar de mejor manera el ancho de banda del internet al evitarse enviar datos de muestreo periódicamente que necesariamente no han cambiado. Después el Gateway enviará mediante protocolo MQTT al broker y el broker realizará una solicitud HTTP desplegando el microservicio para almacenar el componente que corresponda.

Para el caso de los actuadores la capa lógica será la encargada de comandarlos dado que dentro de ella tiene una orquestación y esta es la encargada de realizar el control del proceso. La capa lógica enviará en formato JSON al Gateway de la capa física la información de si se necesita activar o desactivar un actuador, el PLC entonces obedecerá el mensaje que llegó del Gateway y activará o desactivará el actuador.

Figura 23

Estructura general de la Capa Física



Es necesario mencionar que este diseño pertenece a una implementación física, en el presente proyecto no se realizó una implementación, se realizó una virtualización de sensores y actuadores en un servicio web en la capa lógica su diseño será explicado la próxima capa.

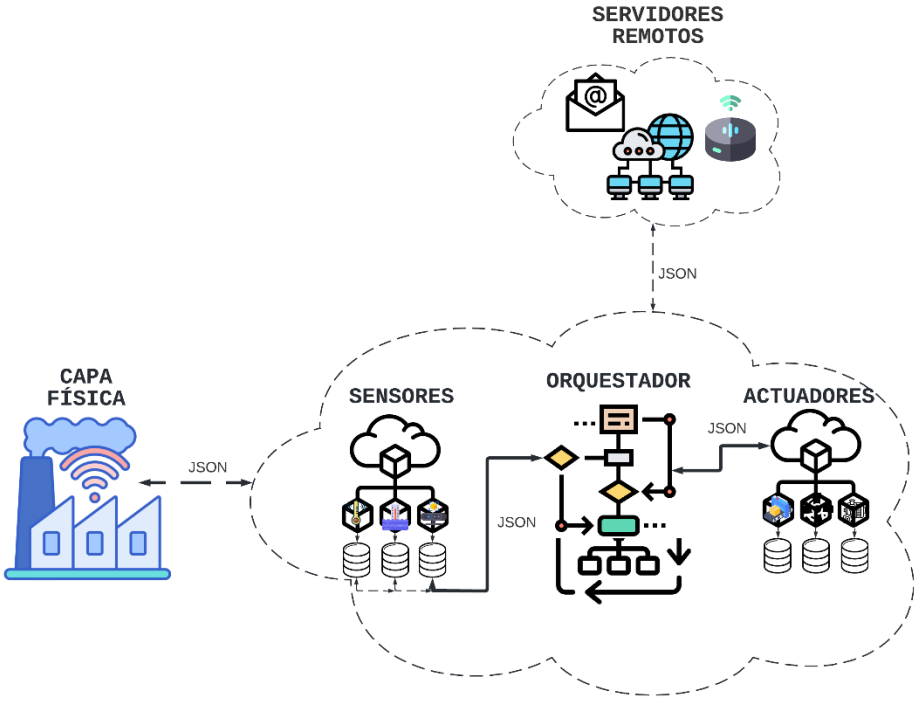
Capa Lógica

Esta capa corresponde el back-end del proyecto, en esta se encuentra toda la lógica de funcionamiento de la fábrica en esta capa se encuentran los microservicios de cada componente ciberfísico, la base de datos, el orquestador y la conexión a servidores remotos como el correo electrónico, en la Figura 24 se puede observar que la capa física envía los datos a la capa lógica, se tienen los microservicios de cada sensor y son coordinados para

efectuar el control y este se ve reflejado en los actuadores adicional se tiene conexión con servidores remotos, se utilizada el protocolo HTTP y el formato de intercambio de datos JSON.

Figura 24

Arquitectura Genérica para Capa Lógica



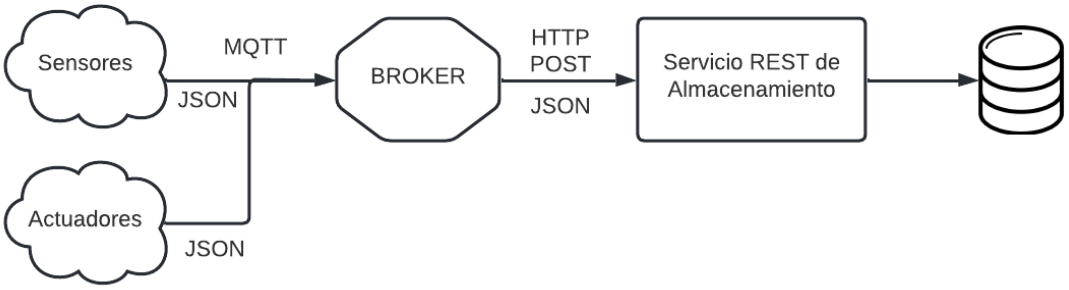
Diseño de microservicios

El presente trabajo debe realizar una arquitectura enfocada a servicios basada en la Web Of Things, por lo cual se utilizan microservicios restfull los cuales utilicen el protocolo HTTP y formato de intercambio JSON, el diseño de los microservicios consta de pequeños Servicios Web que reciban la información de la Thing Description de los componentes ciberfísicos que se encuentran a nivel de campo en la capa física a través del Gateway y son enviados a la capa lógica, mediante peticiones HTTP sea capaz de almacenar la información o de desplegar la información según la URL y el método HTTP utilizado.

Como se observa en la Figura 25 el Broker está suscrito a los tópicos de los componentes ciberfísicos es decir los sensores y actuadores que se encuentran en la capa física, cada vez que estos envíen un mensaje mediante el protocolo MQTT el broker realizará una petición de tipo POST y un servicio realizará el almacenamiento correspondiente según sea el sensor o actuador en su respectiva tabla en la base de datos, el esquema entonces de la figura 25 corresponde a un microservicio que recibe un mensaje JSON mediante una petición HTTP con el método POST y realiza el almacenamiento de ese sensor o actuador.

Figura 25

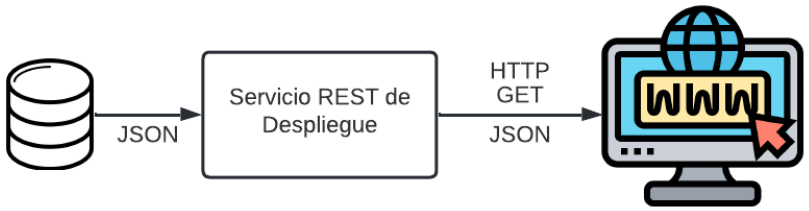
Microservicios Restfull para almacenamiento de componentes ciberfísicos



Mientras tanto, para el diseño de los microservicios de despliegue de la información de los componentes ciberfísicos al realizar una petición de tipo GET al URL correspondiente de cada componente ciberfísico el microservicio del componente desplegará dos opciones: la base de datos entera, o el último valor guardado en la base de datos, de esta manera cada URL corresponderá a un solo componente ciberfísico y cada uno tendrá su propio microservicio independiente del resto de componentes web de la fábrica.

Figura 26

Microservicios Restfull para despliegue de información de componentes ciberfísicos



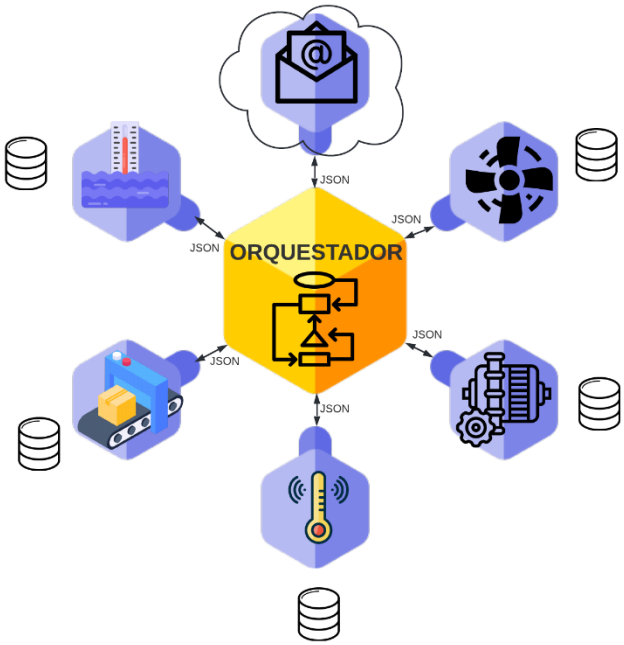
Diseño de orquestador

El orquestador en una Smart Factory se encarga de mediante la coordinación de servicios web realizar los requerimientos del control de procesos y secuencias de los componentes industriales que conforman la fábrica como sensores y actuadores, acciones que antes se llevaban a cabo en nivel local con los controladores y que ahora pasa a ejecutarse este control en la nube como se observa en la Figura 27, el orquestador es el elemento central de los componentes ciberfísicos de la fábrica este mediante un paradigma de solicitud-respuesta es encargado de gestionar el proceso productivo.

El orquestador detectará cambios en las variables de los sensores que son vistos por el orquestador como componentes web pues están descritos en JSON, y será capaz de controlar los actuadores que inciden físicamente en el proceso para dar rumbo a la cadena productiva, siguiendo un flujo de proceso, la orquestación en la industria trae beneficios como la producción flexible y permite integrar los servicios a servidores remotos como el correo electrónico, el orquestador tendrá toda la lógica que seguirá la fábrica cuando activar actuadores, desactivar mediante la interacción de los componentes de la fábrica se puede realizar una producción flexible y orientada a las necesidades del cliente, el orquestador tiene conexión a servidores remotos y podrá decidir cuándo enviar mensajes de correo que pueden ser alarmas o reportes de la producción.

Figura 27

Diseño de orquestador de microservicios



Diseño de componentes ciberfísicos virtuales

En el presente proyecto se diseñaron componentes ciberfísicos que serán virtuales ya que no serán implementados físicamente, serán emulaciones de un servicio web el cual simulará el comportamiento de los elementos de nivel de campo es decir sensores actuadores, estos servicios web de emulación contarán con instrucciones con sentencias de lenguaje de programación JavaScript que para sensores analógicos simularán valores que cambien respecto al tiempo autoincrementándose o decrementándose, mientras que para sensores digitales su activación solo corresponderá a un valor binario, para actuadores neumáticos se realiza la simulación del comportamiento de un cilindro real el cual cuenta con sensores de retracción y extensión, en la Figura 28 se puede observar que los sensores pueden ser simulados por un servicio web alojado en la nube el cual genera información del tipo de componente que este emule.

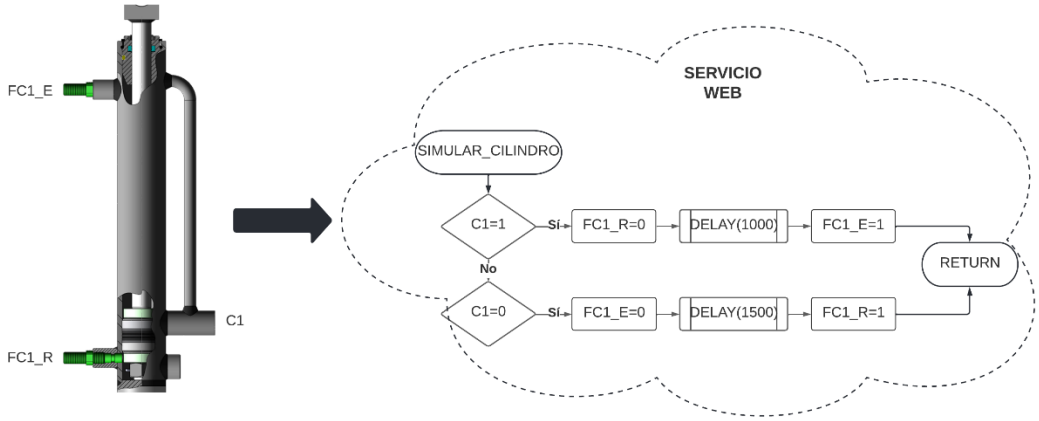
Figura 28

Componentes ciberfísicos emulados por servicios web

A manera de ejemplo en la Figura 29 se muestra la emulación de actuador, en este caso un cilindro doble efecto, a la izquierda podemos ver que el cilindro dispone de dos sensores: FC1_E que corresponde al fin de carrera cuando el cilindro se encuentra extendido, y FC1_R cuando se encuentra retraído y la activación es mediante C1 que corresponde a la alimentación del aire a presión del cilindro, como se puede observar en lo que representaría la lógica del servicio web a emular y esta representado mediante un diagrama de flujo cuando exista aire a presión en el cilindro, es decir C1 se encuentre activado el fin de carrera retraído pasa a desactivarse espera un cierto tiempo y activa el sensor fin de carrera extendido con esto se estaría logrando el objetivo de simulación de un cilindro el cual tiene el comportamiento que debería tener un cilindro doble efecto en la realidad en la que cuando un cilindro se extiende el fin de carrera retraído se desactiva y se activa después de unos segundos el sensor de extendido dependiendo la longitud del cilindro y la presión del aire.

Figura 29

Emulación de un cilindro neumático doble efecto mediante servicios web



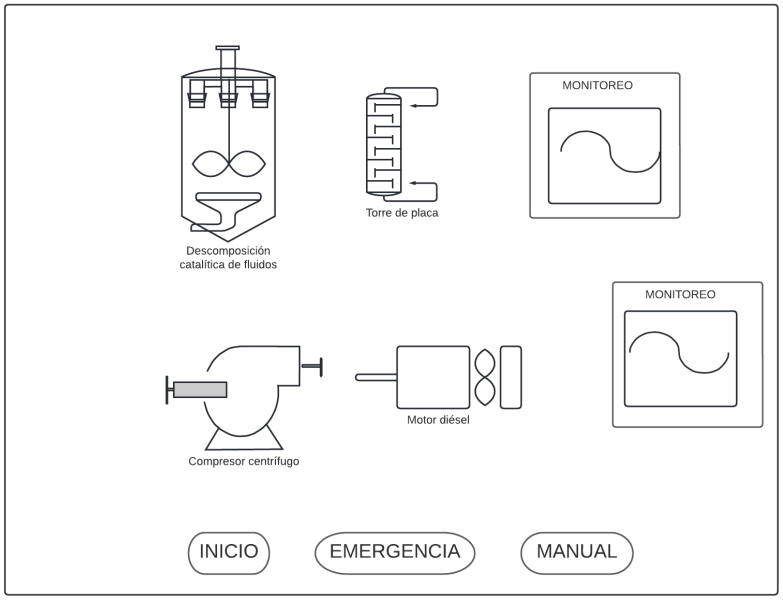
Capa de Aplicación

La capa de aplicación pertenece al front-end de la arquitectura, el diseño de la capa de aplicación consta de la interfaz donde el operador de la fábrica puede monitorear todos los componentes ciberfísicos, ver el proceso completo de la fábrica en tiempo real mediante gráficos históricos y luces piloto, además puede ingresar ordenes de producción flexibles, esta capa es capaz de interactuar con el operador mostrar alarmas y permite tener el control de los modos que todo proceso de automatización debe tener como es el manual, el automático y el de emergencia, además que puede ser desplegada desde cualquier lugar sin importar en que lugar remoto se encuentre el servidor, los datos que llegarán a esta capa serán en formato JSON, esta capa solo podrá recibir datos y enviar datos de sus botones para la interacción con el proceso, no podrá almacenar los datos localmente debido a que los datos se deben encontrar en el servidor, en la Figura 30 se muestra un diseño genérico de una interfáz de una fábrica inteligente en lo primordial está el monitoreo, las gráficas históricas de los componentes y que permita controlar y poner en marcha mediante botones la automatización.

La interfáz debe estar suscrita a todos los tópicos de los componentes ciberfísicos y la interfáz envia

Figura 30

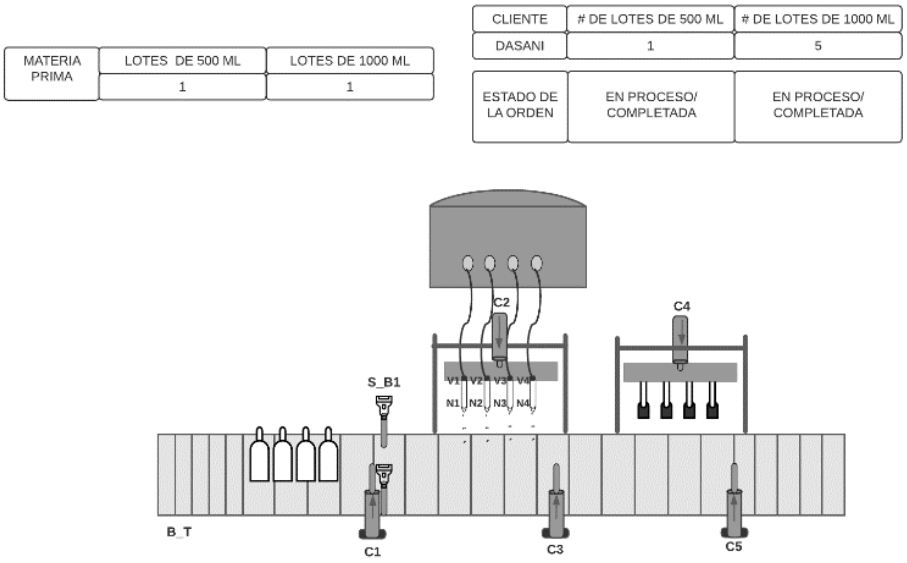
Diseño de capa de aplicación para una Smart Factory



Como se observa en la Figura 31 a modo de ejemplo el diseño de una interfaz la en la cual se puede observar que permite ingresar una orden de producción y permite el monitoreo de todos sus componentes en tiempo real, además que en el diseño es primordial llamar los componentes con acrónimos para poder distinguir entre los elementos que conforman la fábrica. La capa de aplicación puede ser diseñada como una front-end y alojarse en el servidor, pero en este caso por simplicidad y porque el objetivo del proyecto se centra en crear la arquitectura que sea funcional más no en crear una front-end, se utilizó una aplicación para diseño de sistemas SCADA para facilitar la implementación de la interfaz del proyecto.

Figura 31

Diseño de interfaz de ejemplo



Capítulo 4: Implementación

Implementación de la arquitectura

Una vez diseñada capa por capa y teniendo un marco metodológico investigativo se procede a establecer las herramientas y tecnologías ocupadas para cumplir con los requerimientos del proyecto, el cual es implementar una arquitectura de software que basada en la WoT permita crear una herramienta para una fábrica inteligente, en este capítulo se irá mencionando mediante diagramas más detallados, partes de código, librerías y módulos como se llevó a cabo la implementación de la arquitectura en la cual consta la integración de las tecnologías operativas con las tecnologías de la información, el almacenamiento de datos, despliegue de los mismos, configuraciones del broker, servidor y el despliegue de los servicios web utilizados.

A manera de ejemplo se irá presentando diagramas y partes del código mediante los cuales se implementó un caso de estudio para validar la arquitectura y se dará más detalle al final de este capítulo de dicho caso de estudio.

Implementación de los servicios web

Los servicios REST de la arquitectura sin duda son la parte fundamental del proyecto ya que estos son los encargados de almacenar la información de cada componente ciberfísico y permite desplegar la información de cada uno de ellos, son independientes y mediante su coordinación se pueden realizar las tareas de control industrial deseadas en una fábrica inteligente, para la implementación de estos servicios es necesario primero contar con un servidor el cual debe tener un host y un puerto que debe estar accesible a través del internet, por lo cual primero se detalla cómo fue realizada la implementación del servidor mediante la tecnología NodeJS en lenguaje de programación JavaScript.

Implementación del servidor

Para implementar los servicios REST es necesario primero levantar el servidor para lo cual se utilizó NodeJS y el módulo llamado Express, este módulo permite gestionar las peticiones de diferentes métodos HTTP en diferentes rutas URL, establecer el host y que puerto se usará para escuchar las peticiones como se observa en la figura 32 el puerto mediante el cual se va a escuchar las peticiones es el puerto 3000 y el servidor es configurado para que reciba y envié todos los datos en el formato de intercambio JSON, el proyecto se va a dividir en tres archivos sensores.js, actuadores.js y orquestador.js por lo cual también se definen los Routers los cuales son las reglas mediante las cuales Express sabrá como procesar las diferentes peticiones según el archivo en el que se encuentren los servicios.

Figura 32

Implementación del servidor en Node.js

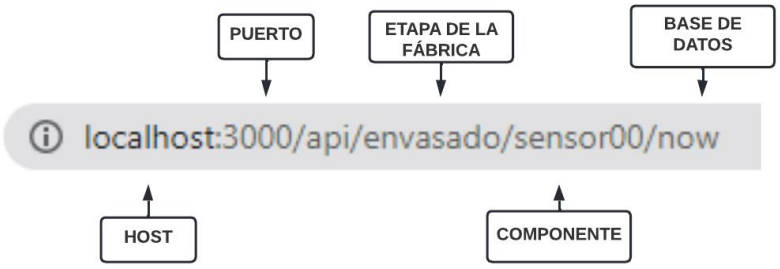
```
const express = require('express');
const app = express();
const morgan = require('morgan');

//Configuraciones
app.set('port', process.env.PORT || 3000);
app.set('json spaces', 2);

app.use(express.urlencoded({extended: false}));
app.use(express.json());

//routes
app.use('/api/envasado',require('./routes/sensores'))
app.use('/api/envasado',require('./routes/actuadores'))
app.use('/api/envasado',require('./routes/orquestador'))

// Empezando el servidor
app.listen(app.get('port'), () => {
    console.log(`Server on port ${app.get('port')}`);
    });
```

Implementación de los servicios REST de los componentes

Para la implementación de los servicios REST de los componentes ciberfísicos se define la URI que deberá tener cada componente, esta URI es única para cada uno de ellos en esta se realizarán las peticiones, la URI será la encargada de localizar los recursos y mediante el servicio web responde con el recurso solicitado, en la figura 33 se observa cómo es el modelado para la implementación de la URL de cada componente, el host, el puerto, la etapa de la fábrica, el componente y se puede elegir entre el último ítem guardado en la base de datos o toda la base de datos.

Figura 33

Modelado de la URI de los componentes ciberfísicos



Una vez definido como se implementará la URI de cada componente se procede a crear el servicio web mediante instrucciones de JavaScript se tiene dos métodos POST y GET, el primero almacenará la información de cada uno de los componentes y el segundo desplegará esa información.

Servicios Web para almacenamiento de recursos de componentes

Se sabe entonces que la información debe ser en formato JSON con base a la Thing Description por lo cual, mediante una solicitud HTTP con el método POST a una URI como la del ejemplo de la figura 34 almacenaría en la tabla del componente correspondiente que se encuentre en la base de datos, este principio y código se usa para cada uno de los componentes ciberfísicos solo cambia la URI de cada componente, y el ítem “id” proporcionado

en JSON debe ser único para que la base de datos encuentre en que tabla debe ser guardado dicho componente.

Figura 34

Implementación de servicio web para almacenamiento de componente ciberfísico

```
router.post('/sensor00', (req,res) =>{
    const {id, codigo, magnitud, modelo, estado, ubicacion, observaciones,
unidad, variable}= req.body;
    const query= `
        CALL env_sensores_ADD(?, ?, ?, ?, ?, ?, ?, ?, ?);
    `;
    mysqlConnection.query(query,[id, codigo, magnitud, modelo, estado,
ubicacion, observaciones, unidad, variable],(err,rows,fields) =>{
        if(!err){
            res.json({Status:'Sensor Guardado'});
        }
        else{
            console.log(err);}
        });
    });
```

Servicios Web para despliegue de recursos de componentes

Para poder desplegar la información que a sido guardada de los componentes ciberfísicos de la fábrica se utilizan servicios web de igual manera los cuales mediante el método HTTP GET deberá desplegar la información de la base de datos del recurso del componente ingresado en la URI, este método puede ser un navegador web en el cual se desee observar la información de algún componente de la fábrica.

En la Figura 35 se puede observar porciones de código del caso de estudio en la cual la primera porción responde al servicio web si se desea obtener el último valor que se guardó en la base de datos del componente ciberfísico en cuestión, este servicio web busca en la tabla el componente y lo responde con el último ítem guardado.

En la segunda porción de código se despliega toda la base de datos, este servicio es útil para realizar las gráficas históricas, analizar datos y consultar valores pasados del

componente ciberfísico, esta idea y porción de código se extiende para todos los componentes ciberfísicos de la fábrica inteligente.

Figura 35

Servicio web para el despliegue de la información de los componentes

```
//Obtiene ultimo valor de la base de datos
router.get('/sensor00/now', (req,res) =>{
    mysqlConnection.query('SELECT * FROM env_s00 WHERE id = (SELECT
MAX(id)  FROM env_s00)',(err,rows,fields) =>{
        if(!err){
            res.json(rows);}
        else{
            console.log(err);}
            });
    });
//Obtiene toda la base de datos
router.get('/sensor00', (req,res) =>{
    mysqlConnection.query('SELECT * FROM env_s00',(err,rows,fields) =>{
        if(!err){
            res.json(rows);}
        else{
            console.log(err);}
        });
    });
```

Implementación de la base de datos de los servicios

Los servicios web se encargarán de gestionar las solicitudes HTTP que lleguen al servidor y podrán realizar el almacenamiento o despliegue de la información de los componentes se necesita un lugar para guardar esta información para esto es necesario la utilización de una base de datos se utilizó MySQL Workbench por lo cual fue necesario un host, un usuario, una contraseña para mantener seguros los datos, el nombre de la base de datos y la librería de NodeJS que hace posible la conexión entre ambas partes como se observa en la figura 36 se crea una instancia para poder ocuparla en todos los archivos sensores, actuadores y orquestador pues los tres archivos son los encargados de almacenar los componentes

ciberfísicos de la fábrica dentro de las tablas de la base de datos, es importante recalcar que la base de datos se encontrará junto al servidor aunque también podría colocarse de manera remota distanciada del servidor, en la figura 37 se observa la base de datos creada en MySQL.

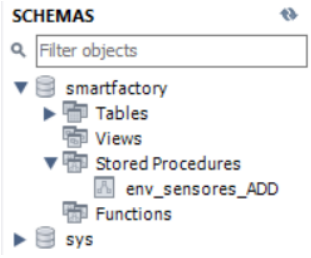
Figura 36

Configuración de la base de datos en el servidor

```
const mysql = require('mysql');
const mysqlConnection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password:'Liguista.11',
    database: 'SMARTFACTORY',
    multipleStatements: true
});
mysqlConnection.connect(function (err) {
    if(err){
        console.log(err);
        return;
    }   else{
        console.log('Db is connected');
    }

});
module.exports = mysqlConnection;
```

Figura 37

Base de datos de la Smart Factory



Una vez que la base de datos se encuentra conectada al servidor es necesario crear un procedimiento el cual será el encargado de instanciar las variables de la Thing Description de los componentes según el tipo de dato: INT, VARCHAR, float, etc.

Este procedimiento será llamado en los servicios Rest antes explicados, es decir el servicio web recogerá el JSON y enviará el JSON en variables al procedimiento, el cual mediante instrucciones condicionales dirigirá según una variable llamada “código” la cual es única de cada componente ciberfísico, y guardará en la tabla respectiva a la que pertenezca esa información como se observa en la figura 38.

Figura 38

Procedimiento de la base de datos MySQL

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `env_sensores_ADD`(
      IN _id INT,
      IN _codigo VARCHAR(20),
      IN _magnitud VARCHAR(50),
      IN _modelo VARCHAR(20),
      IN _estado VARCHAR(20),
      IN _ubicacion VARCHAR(20),
      IN _observaciones VARCHAR(20),
      IN _unidad VARCHAR(20),
      IN _variable float(40)
)
BEGIN
      IF _codigo = 'env_s00' THEN
            INSERT INTO env_s00
(codigo,magnitud,modelo,estado,ubicacion,observaciones,unidad,variable)
            VALUES
(_codigo,_magnitud,_modelo,_estado,_ubicacion,_observaciones,_unidad,_variab
le);
            SET _id = last_insert_id();
      ELSEIF _codigo = 'env_s01' THEN
            INSERT INTO env_s01
(codigo,magnitud,modelo,estado,ubicacion,observaciones,unidad,variable)
            VALUES
(_codigo,_magnitud,_modelo,_estado,_ubicacion,_observaciones,_unidad,_variab
le);
            SET _id = last_insert_id();
      ELSEIF _codigo = 'env_s02' THEN
```

```
            INSERT INTO env_s02
(codigo,magnitud,modelo,estado,ubicacion,observaciones,unidad,variable)
            VALUES
(_codigo,_magnitud,_modelo,_estado,_ubicacion,_observaciones,_unidad,_variab
le);
            SET _id = last_insert_id();
```

Una vez creado el procedimiento se puede observar que cada componente se guarda en su tabla en particular en la figura 39.

Figura 39

Tabla de componente ciberfísico

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	codigo	magnitud	modelo	estado	ubicacion	observaciones	unidad	variable
1	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0
2	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0
3	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	1
4	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0
5	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	1
6	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0
7	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0
8	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	1
9	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0
10	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	1
11	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0
12	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0
13	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	1
14	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0
15	env_act07	Electrovalvula C4	Generico	Nuevo	Ec,Sangolqui	Ninguna	Booleano	0

Implementación de la integración del correo electrónico

El correo electrónico es una de las características principales de la capa lógica para demostrar la integración del proyecto con servidores remotos de cualquier tipo, el correo electrónico será el encargado de notificar mediante mensajes electrónicos como se encuentra el ciclo de producción, si surgieron alertas o si las órdenes fueron completadas del cliente, para implementar esta característica se utilizó NodeMailer el cual es un módulo de NodeJs que permite realizar solicitudes Rest a servidores remotos IMAP y SMTP, asegura un envío seguro mediante TLS.

En la figura 40 se puede observar la configuración que se realiza con NodeMailer como es el host y el puerto en este caso Hotmail, el ingreso del correo que vamos a utilizar, la contraseña y activamos la seguridad TLS, de esta manera estaría configurado el servidor remoto de correo electrónico y el orquestador será capaz de enviar mensajes electrónicos a gerentes, proveedores y clientes que sean notificados sobre el ciclo productivo de la fábrica.

Figura 40

Configuración de NodeMailer

```
const transporter = nodemailer.createTransport({
  host: "smtp.gmail.com",
  port: 465,
  secure: true, // true for 465, false for other ports
  auth: {
    user: '11ivanacu@gmail.com', // generated ethereal user
    pass: 'password', // generated ethereal password
  },
  tls:{
    rejectUnauthorized: false
  }
});
```

Implementación de los servicios de emulación de los componentes virtuales

Como se ha venido abordando, en la capa física se encuentran componentes que miden magnitudes físicas del medio, en el presente proyecto no existe una implementación física, por lo tanto, sensores y actuadores son simulaciones realizadas en servicios web de componentes ciberfísicos virtuales.

Los servicios web de simulación son porciones de código que se están ejecutando en el servidor, son realizados en la plataforma de NodeJs en lenguaje de programación JavaScript con instrucciones condicionales, bucles, código sincrónico y más funciones que ofrece el propio lenguaje para de esta manera poder simular el comportamiento que deberían tener los componentes virtuales tal como si se los implementará físicamente como: sensado del llenado

de un líquido, el aumento o sensores para el disminución de temperatura o como el simulado de la activación de una electroválvula o de un cilindro en el caso de actuadores.

En el anterior capítulo se pudo ver como se estructuró el diseño de estos componentes y a grosso modo se vió como un cilindro puede ser una simulación de un servicio web con un diagrama de flujo, ahora para ejemplificar como se realiza la implementación de la emulación de componentes ciberfísicos virtuales se muestra cómo se realizó la implementación de los componentes del caso de estudio en NodeJs los cuales podrían ser reutilizados en cualquier proceso de emulación que implique estos componentes.

Es importante mencionar que las estos servicios web son independientes y funcionan con el paradigma de hilos que nos ofrece Node.js el cual es que se pueden desplegar varios servicios al mismo tiempo, esto sumamente importante al momento de simular pues en la realidad suceden acciones que son en paralelo.

Emulación de actuadores como componentes ciberfísicos virtuales

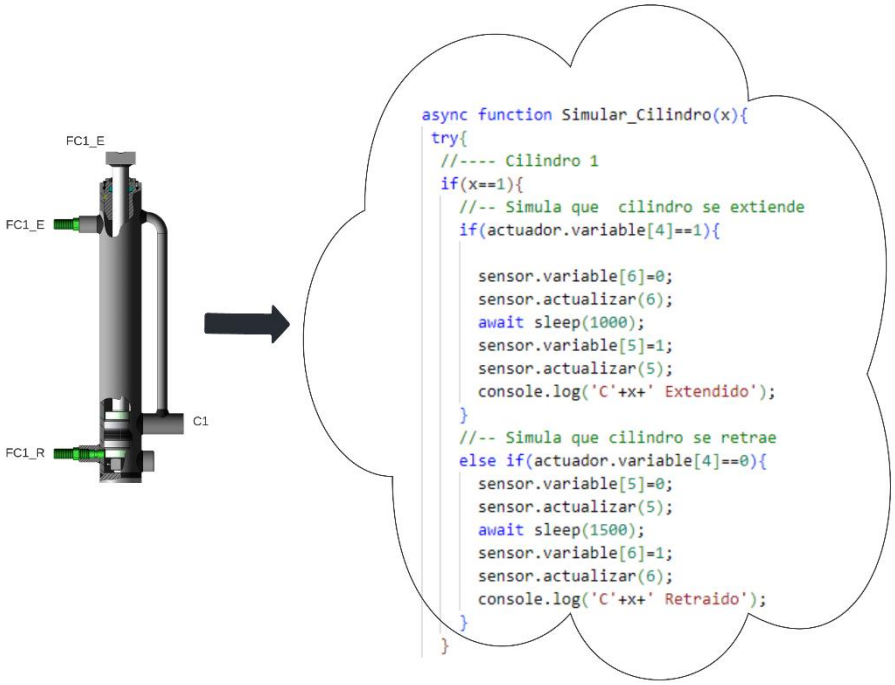
Los actuadores ciberfísicos son dispositivos los cuales interactúan con el entorno que los rodea, la virtualización o emulación de estos dispositivos se realiza cuando es necesario realizar una validación de un sistema y estos no se encuentran físicamente implementados, están formados por servicios web, implementados en lenguaje JavaScript en el entorno de ejecución NodeJs.

En la figura 41 se observa que un cilindro que vendría a ser un actuador neumático el cual, ligado al PLC es un componente ciberfísico, podría ser emulado por un servicio web como se ve a la derecha, el cilindro tiene dos sensores FC1_E y FC1_R cada vez que el actuador se activa esos sensores deberían cambiar su valor por lo cual en su implementación es una función llamada Simular_Cilindro, recibe el cilindro a emular ya que se pueden emular algunos a la vez este pregunta si fue activada la válvula si fuera así desactiva o activa los sensores

fines de carrera según corresponda utilizando await que es una instrucción que espera un tiempo para seguir con la siguiente instrucción, simulando en retardo entre su extensión o retracción.

Figura 41

Implementación emulación de actuador mediante servicios web



Emulación de sensores como componentes ciberfísicos virtuales

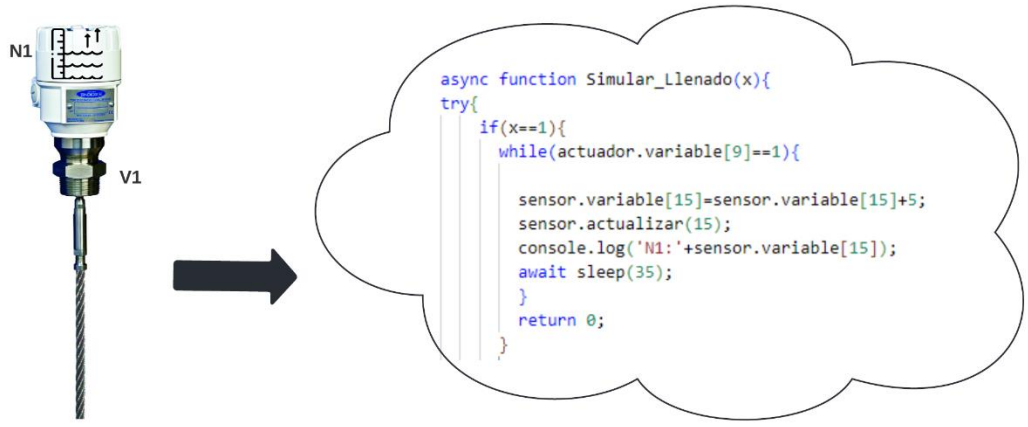
Tal como los actuadores, los sensores de igual manera proporcionan información virtual la cual es utilizada para validar un sistema y están implementadas en un servicio web con JavaScript en un entorno de NodeJS.

La implementación de un sensor virtual se realiza mediante la emulación en un servicio web se ejemplifica esto en la figura 42 a la izquierda se puede observar un sensor de nivel el cual contiene una válvula la cual detiene o permite el paso del líquido y el sensor de nivel como tal que transmite datos análogos en proporción al nivel del líquido que se encuentra sensando a

la derecha se observa su servicio web el cual es una función asíncrona que simula el llenado, su lógica es la siguiente cuando la válvula esté abierta permite cambios incrementales progresivos en el nivel del sensor virtual cada 35 milisegundos, y este se encuentra en un bucle while el cual permite que mientras la válvula esté abierta el código se mantenga ejecutando, si en algún momento la válvula se cierra retorna del servicio web, de este manera mediante NodeJS y lenguaje JavaScript se implementa un sensor de nivel genérico.

Figura 42

Implementación emulación de sensor mediante servicio web



Implementación de la interfaz de la arquitectura

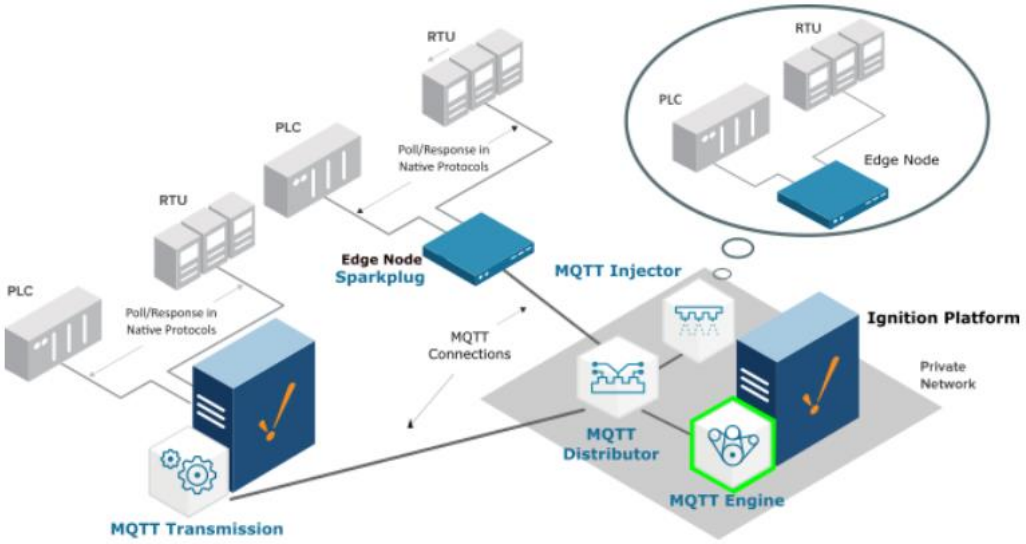
Para la implementación de la capa de aplicación se realizó una interfaz mediante el software Ignition el cual es capaz de suscribirse de manera automática a todo un tópico que puede ser la etapa de producción de una fábrica se encarga de descubrir cada componente y ponerlo a disposición en etiquetas.

Ignition tiene la capacidad para el monitoreo en tiempo real, gráficas históricas, alarmas, tablas y generar reportes, en el presente proyecto, Ignition será únicamente un cliente por lo tanto el encargado de alarmas, orden de producción y demás es el orquestador, como se observa en la figura 43 las tecnologías que hacen esto posible son los módulos de Ignition

MQTT Distributor el cual permite encaminar los mensajes según los tópicos, MQTT Engine que permite recibir los mensajes MQTT, y el módulo MQTT Transmission el cual permite enviar o transmitir como su nombre lo indica mensajes MQTT.

Figura 43

Interfaz Ignition+MQTT

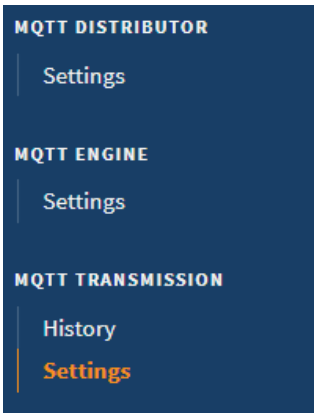


Nota. Figura tomada de (Ignition, 2021).

De esta manera las tecnologías que ayudarán a realizar la interfaz de la capa de aplicación son los módulos antes mencionados cada uno de estos deberá ser configurado de la siguiente manera.

Figura 44

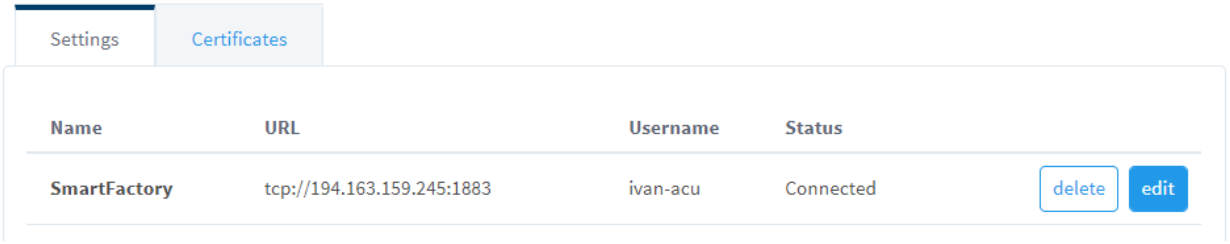
Módulos de Ignition MQTT



En la figura 45 se muestra cómo se realiza la configuración de MQTT Engine el cual está conectado al host donde se encuentra broker y al puerto 1883 que pertenece a Mosquitto Broker que es el Broker utilizado en el proyecto y será el encargado de manejar toda la mensajería ligera MQTT que envíen los componentes ciberfísicos, este módulo es externo a Ignition, pero se puede descargar de su página oficial y se puede agregar a Ignition.

Figura 45

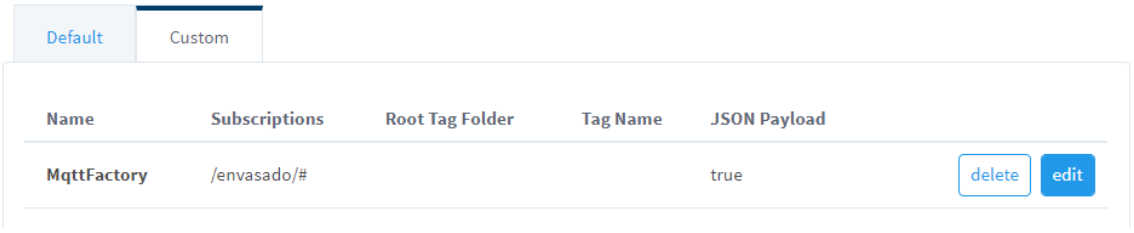
Configuración de MQTT Engine



En la figura 46 se puede observar los tópicos a los que Ignition estará suscrito para realizar el monitoreo de las variables de los componentes ciberfísicos, se puede observar que se usa de ejemplo el caso de estudio en la etapa de envasado y el “#” corresponde a que se suscriba a todos los sensores y actuadores que pertenezcan a esta etapa.

Figura 46

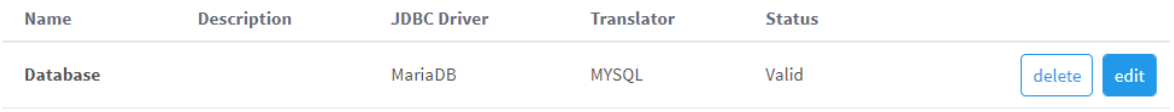
Configuración de tópicos para Ignition



Finalmente se conecta la interfaz a la base de datos esto servirá para poder realizar los gráficos históricos de los datos que sean archivados por los servicios web Rest enviados por los sensores y actuadores.

Figura 47

Conexión Ignition con la base de datos



Con estos módulos Ignition es capaz de reconocer a todos los componentes de la fábrica y estará configurada para crear la interfaz con las herramientas gráficas que proporciona según la necesidad de cada fábrica en torno a requisitos de HMI, modos de funcionamiento, alarmas despliegue de información, etc.

Despliegue de la arquitectura final

Después de detallar la forma de implementación de la arquitectura, tecnologías y herramientas utilizadas para cumplir con los requerimientos de diseño impuestos y los objetivos del proyecto se realiza el despliegue del proyecto desde el back-end hasta el front end.

Mosquitto Broker

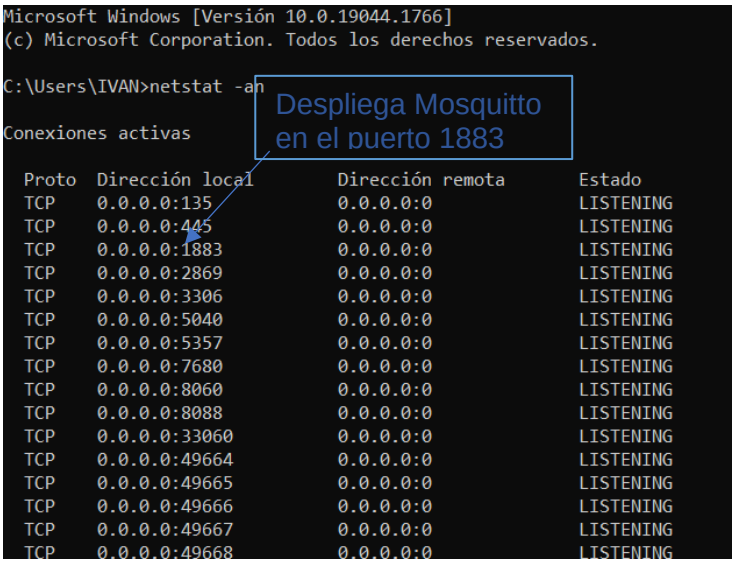
El encargado de gestionar toda la mensajería ligera MQTT es Mosquitto Broker 2.0, este será el encargado de permitir que el proyecto tenga los tópicos necesarios de todos los componentes utiliza el puerto 1883.

Para levantar Mosquitto únicamente es necesario descargar de su página oficial en el servidor remoto o local acceder a la carpeta donde se encuentra instalado y ingresar la siguiente línea de comando, en la figura 48 podemos ver que se encuentra desplegado.

```
mosquitto -c
```

Figura 48

Despliegue y funcionamiento del Broker



Backend

Para el despliegue del backend es necesario replicar lo que se realizó en la computadora local si se desea que se encuentre en un servidor remoto, ejecutar cada uno de los archivos que son de extensión .js, cambiar el host por el cual se vaya a utilizar e igual elegir el puerto que por defecto es el puerto 3000.

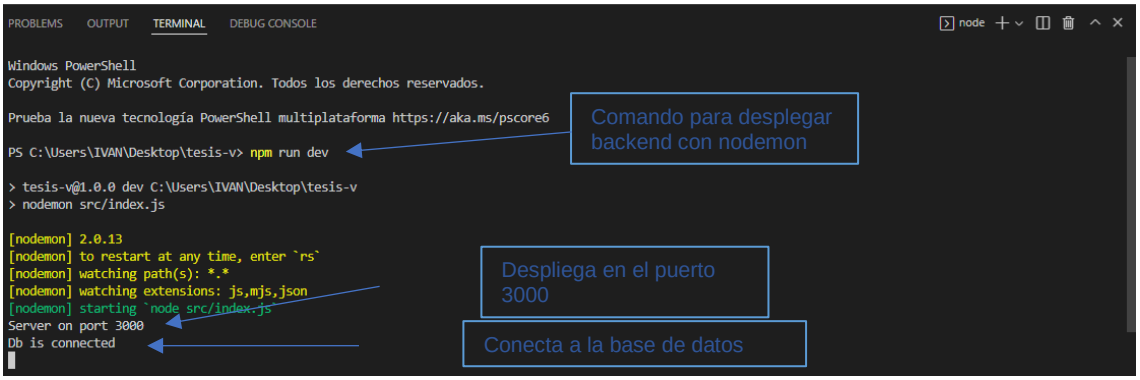
- Instalar Visual Studio Code 1.67.2 o mayor.

- Abrir la carpeta del proyecto en Visual Studio.
- No es necesario ninguna instalación adicional debido a que la carpeta del proyecto ya cuenta con los módulos antes mencionados y las librerías requeridas.
- Cambiar la dirección de MQTT por la del servidor y el puerto se mantiene en el predeterminado con 1883.
- Escribir el comando “npm run dev” en el terminal de Visual Studio Code, este comando es el encargado de correr todos los archivos .js de la carpeta del proyecto.

Una vez realizados todos estos pasos el servidor estará levantado y los servicios web funcionando.

Figura 49

Despliegue de backend del proyecto

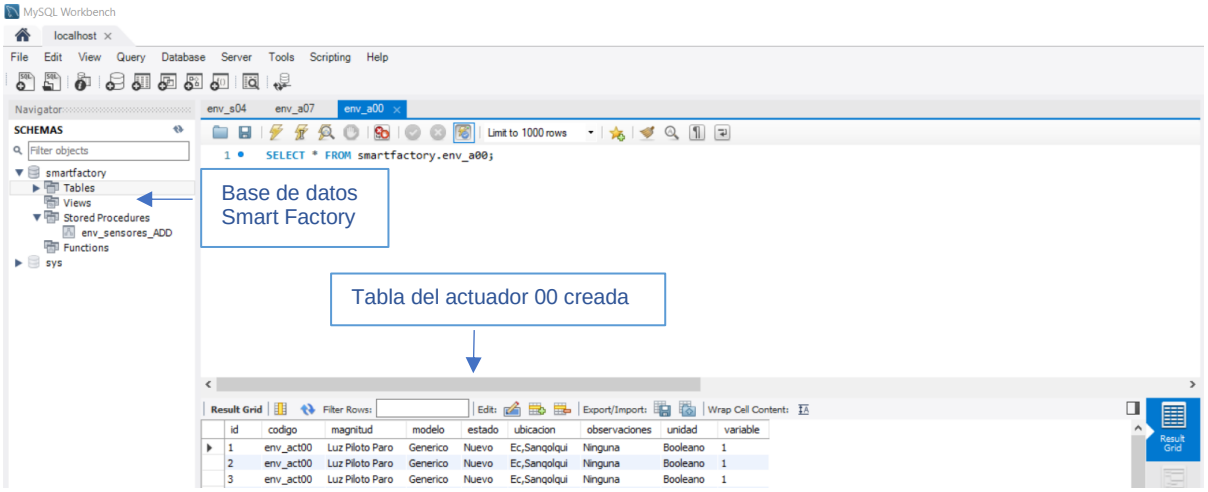


Base de datos

Para la base de dato se utilizó MySQL Workbench 8.0, tan solo es necesario descargar el programa de la página oficial crear una base de datos con cualquier nombre, crear un nuevo procedimiento y ejecutar el procedimiento que se mostró anteriormente esto creará las tablas y las direccionará cada componente de cada servicio web a una tabla, la base de datos de MySQL utiliza el puerto por defecto 3306.

Figura 50

Despliegue de la base de datos



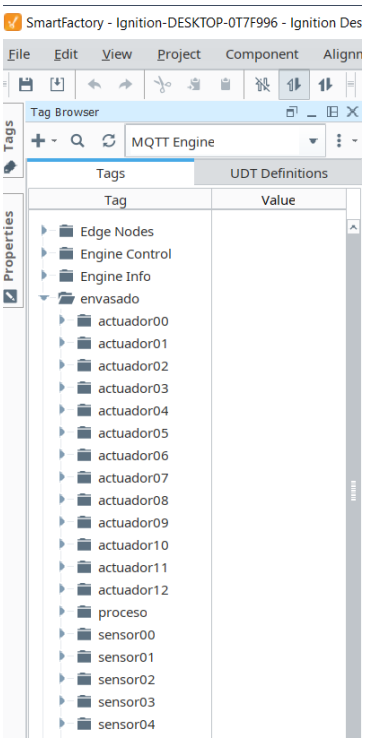
Ignition

Para desplegar Ignition es necesario descargar de su página oficial e instarlo, la interfaz es creada en base al diseño del programador y de sus objetivos con la interfaz ya que ofrece muchas funcionalidades, a continuación, se muestran los pasos para desplegarlo en cualquier computadora, el puerto que utiliza Ignition por defecto es el 8088.

- Descargar e instalar Ignition e ingresar a http://localhost:8088/web.
- En Configuración instalar los módulos MQTT Engine y MQTT Transmissor.
- Configurarlos como se observó anteriormente en implementación con el host en el que se encuentre el cliente.
- Conectar Ignition con el broker MQTT y el puerto que se vaya a utilizar.
- En los tópicos agregar la etapa de la fábrica por ejemplo /envasado/#, Ignition detectará automáticamente los componentes que esa etapa posea.
- Conectar Ignition con la base de datos.
- Abrir Desing Luncher de Ignition.
- Importar el proyecto que se encuentra en anexos.

Figura 51

Despliegue de Ignition



De esta manera se desplegaría todo el proyecto para adaptar el proyecto en un uso de cualquier fábrica.

Capítulo 5: Pruebas de validación

Implementación del escenario de validación

Para realizar una validación de la arquitectura propuesta se realizó un escenario de prueba o caso de estudio el cual es la etapa de una fábrica embotelladora como se había expuesto en el capítulo 1, se utilizará como caso de estudio la etapa de envasado la cual consta de 3 estaciones, transporte de botellas, llenado y taponado de las botellas, todo esto se hará utilizando el diseño e implementación que se ha mencionado en los anteriores capítulos para validar que la arquitectura propuesta es funcional, se utilizarán herramientas y tecnologías como son los servicios web que virtualizan los componentes ciber físicos existentes del caso de estudio, servicios Rest y el frontend o capa de aplicación es realizado en Ignition.

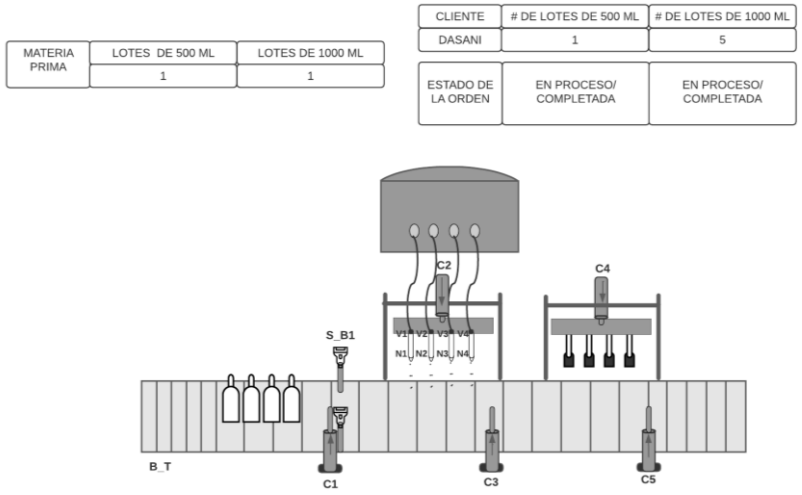
El caso de estudio deberá incorporar todo el análisis para el diseño e implementación que se ha venido desarrollando a lo largo de todo el presente escrito, además la lógica de la fábrica en la nube en el orquestador es un caso particular para cada fábrica por lo cual el orquestador deberá cumplir algunas condiciones que se realizaron para implementar el caso de estudio.

- La producción debe ser flexible y acorde a lo que el cliente solicite, en este caso de etapa de envasado tener al menos dos tipos de botellas de diferentes tamaños.
- Se puede ingresar la materia prima en este caso botellas para ser llenadas.
- La automatización debe constar de al menos 3 modos de operación: automático, manual y emergencia.
- Modo automático: al agregar la orden de producción debe funcionar de forma autómata.
- Modo manual: se puede ingresar desde cualquier punto del proceso, para la producción y permite activar o desactivar cualquier actuador de la fábrica.

- Modo emergencia: debe ser capaz de parar la producción en marcha, activar un protocolo de seguridad, como apagar todos los actuadores.
- Debe poseer alarmas de: sobre llenado, falta de materia prima.
- Conectarse a un servidor remoto como el correo electrónico para reportar estados de la cadena de producción como: producción lista al gerente de la fábrica y al cliente todo esto desplegando un reporte de la producción que está lista, y si falta materia prima al proveedor de materia prima indicando que tipo de botellas hay escasez.
- Monitoreo en tiempo real de todas los componentes web del proceso en la interfaz.
- Debe constar de sensores análogos y digitales.
- Visualizar y editar cualquier atributo de la Thing Description.

Figura 52

Implementación de caso de estudio

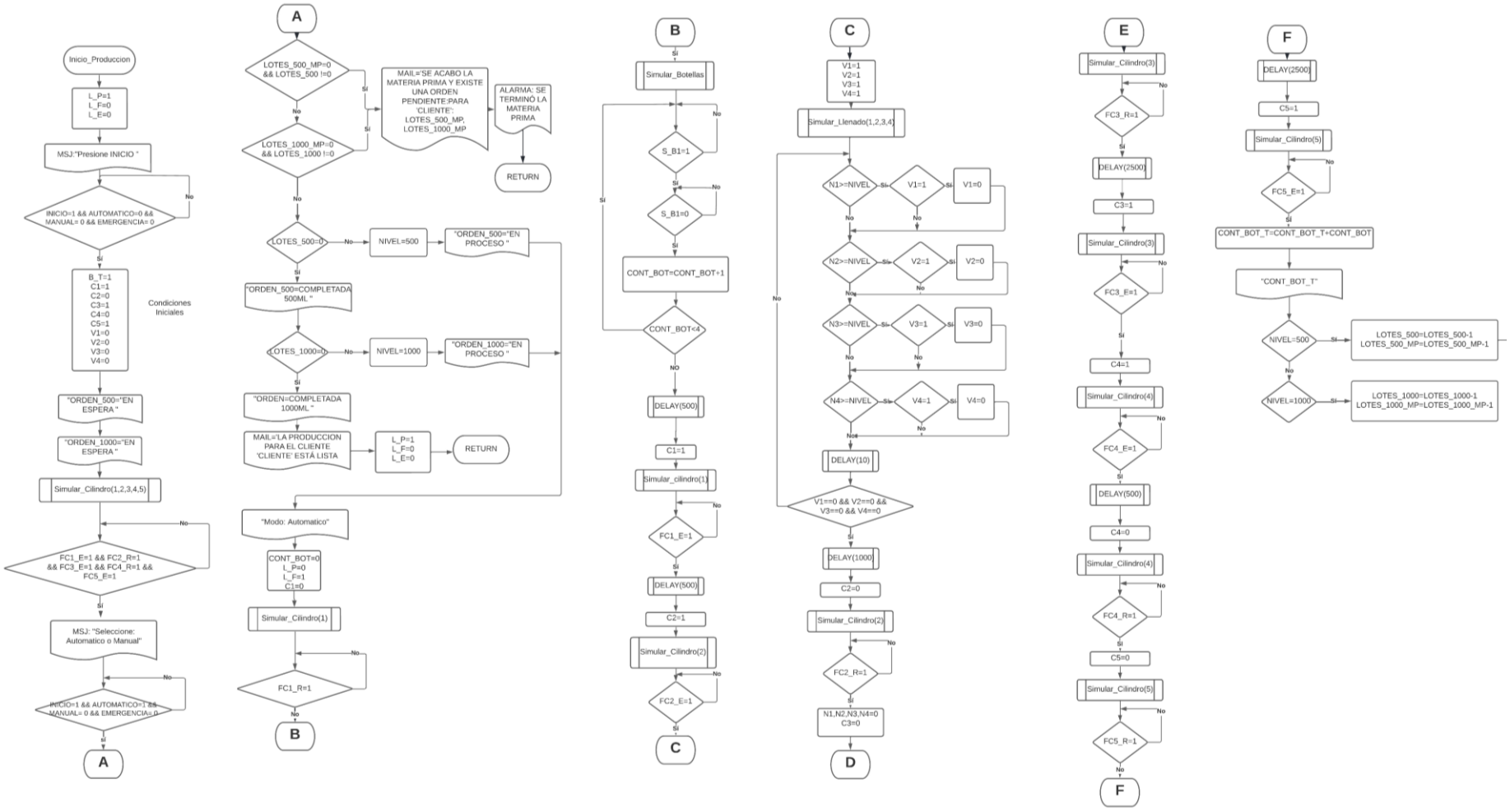


En la figura 53 se puede observar el diagrama de flujo que representa la lógica con la que funciona el orquestador del caso de estudio, está encargado de todo el control industrial de la fábrica inteligente con todos los sensores y actuadores interactuando para un fin común el

cual es realizar un proceso productivo flexible y que se encuentre en la nube, cabe destacar que esta implementación del backend del proyecto en la cual también se encuentra el orquestador, fue trasladada a un servidor remoto para su validación y Ignition actuó de cliente en una pc local, el servidor albergará todos los servicios web y además también incorporará la base de datos con los procedimientos y tablas de cada componente web de la fábrica.

Figura 53

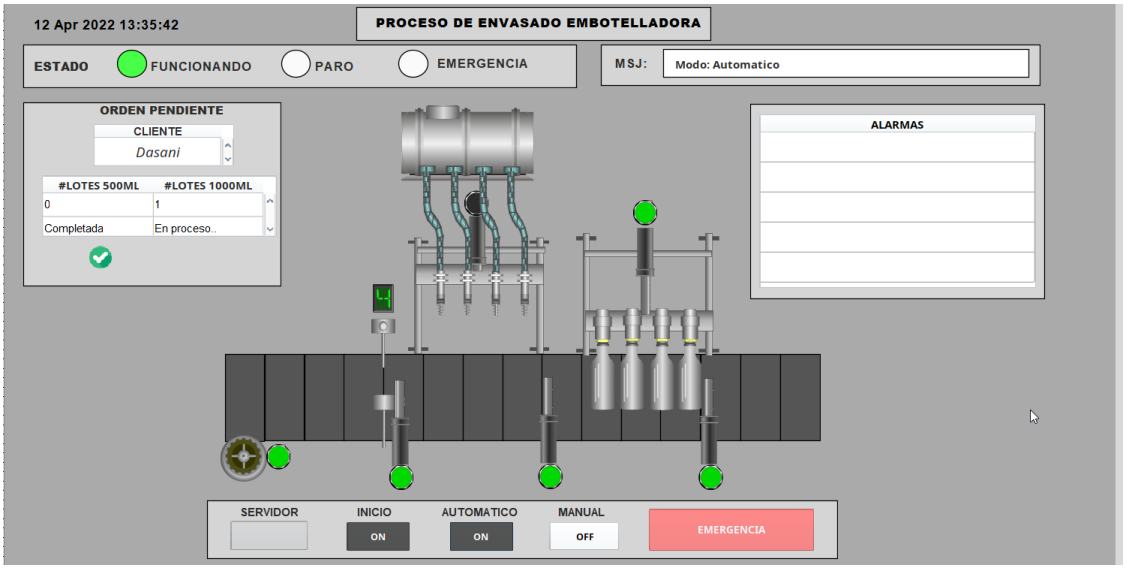
Diagrama de flujo de funcionamiento de orquestador



En la Figura 54 se puede observar la interfaz que fue creada en Ignition es decir la capa de aplicación de la arquitectura planteada para el caso de estudio en cuestión, la cual consta de luces pilotos, formulario para ingreso de cliente y lotes, alarmas y mensaje interactivo para mejor entendimiento del usuario, botones de modos incluyendo emergencia, el botón de servidor despliega el orquestador, cabe recalcar que cada uno de los componentes ciber físicos posee una propia ventana a estilo popup en la cual muestra gráficos históricos y permite visualizar y modificar la Thing Description de cada componente de igual manera el proceso se observa de forma interactiva es decir se puede observar el conteo movimiento de las botellas y los cilindros neumáticos, llenado y taponado de cada lote de 4 botellas, para el caso de estudio se utilizaron nombres genéricos que ayudan a que este caso de Smart Factory sea utilizado en cualquier aplicación o etapa de una fábrica con todo el proceso de diseño e implementación que se ha venido abordando en el escrito del presente proyecto.

Figura 54

Interfaz de caso de estudio

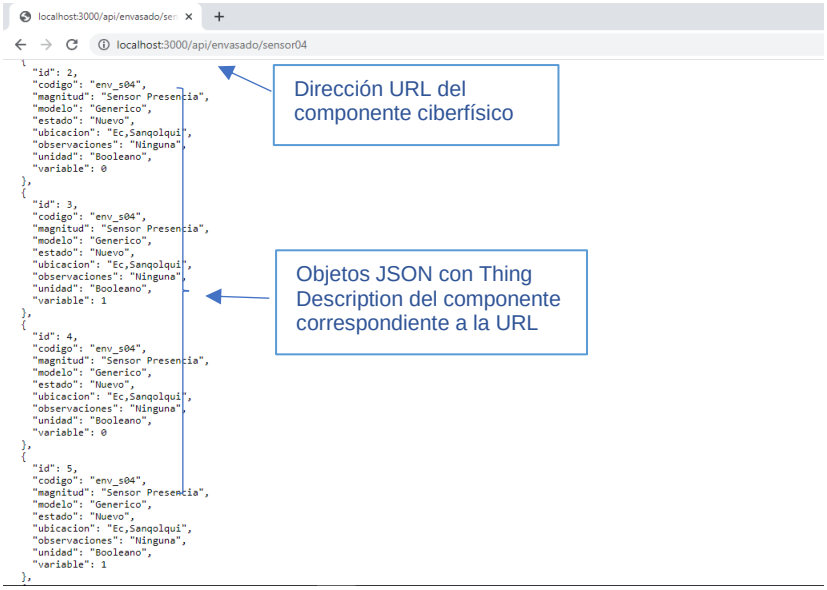


En la figura 55 se observa el despliegue de la información en formato JSON de un componente ciber físico en la cual se puede verificar el correcto funcionamiento de los

microservicios de despliegue, mediante un navegador que realiza métodos GET a las direcciones URL de la arquitectura se puede recuperar la información de cualquier componente del caso de estudio de la Smart Factory.

Figura 55

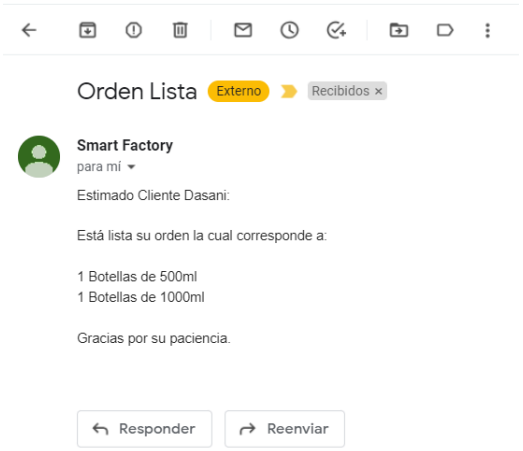
Despliegue de información en JSON de componentes ciber físicos.



Además, el orquestador coordina acciones también en servidores remotos como ejemplo el correo electrónico como se puede apreciar en la Figura 56 cuando la producción se encuentra lista el orquestador ordena que se envíe correos electrónicos al gerente y al cliente, en caso de terminarse la materia prima también se podría enviar correos electrónicos a proveedores para que abastezcan a la fábrica de estos productos lo más pronto posible, una ventaja importante es que el orquestador tiene los datos de todos sus componentes y el correo puede ser personalizado con datos en tiempo real que son recuperados de la cadena de producción.

Figura 56

Verificación de funcionamiento de conexión con servidores remotos



Pruebas del sistema

Una vez se comprueba que la arquitectura funciona y cumple con los requisitos de diseño es importante realizar pruebas de carga para lo cual se utiliza la herramienta de software Gatling la cual estresa el sistema para evaluar su correcto funcionamiento, Gatling permite crear un script en la IDE IntelliJ IDEA con lenguaje SCALA en el cual se configuran todas las peticiones posibles del caso de estudio.

El código fue creado en la IDE antes mencionada con el lenguaje SCALA, para mejor entendimiento se divide el código en 4 partes, la primera parte contiene la importación de paquetes necesarios para las definiciones de instrucciones de lenguaje Gatling:

```
package Simulaciones
import io.gatling.core.scenario.Simulation
import io.gatling.core.Predef._
import io.gatling.http.Predef._
import scala.concurrent.duration.DurationInt
```

La segunda parte consta de definir el protocolo a utilizar en este caso HTTP, la URL base, la cabecera y el tipo de dato que el servidor deberá entregarnos como en este caso son datos en formato JSON.

```
val httpConf = http.baseUrl(url="http://localhost:3000")
  .header(name="Accept", value="application/json")
  .header(name="content-type", value="application/json")
```

La tercera parte consta de definir el escenario, el cual es el camino en que un usuario realizará en la navegación de las URL realizando peticiones GET, consta de su método GET, su URL y que respuesta debe entregar en este caso el código 200 que significa satisfactorio o con éxito, en el siguiente código se visualiza solo una porción de código ya que son 38 diferentes peticiones, pero tienen la misma estructura.

```
val scn =
  scenario(scenarioName = "Prueba de funcionamiento")
    .exec(
      http(requestName = "Sensor00 histórico")
        .get("/api/envasado/sensor00")
        .check(status is 200))


    .exec(
      http(requestName = "Sensor00 actual")
        .get("/api/envasado/sensor00/now")
        .check(status is 200))


    .exec(
      http(requestName = "Sensor01 histórico")
        .get("/api/envasado/sensor02")
        .check(status is 200))

    .exec(
      http(requestName = "Sensor01 actual")
        .get("/api/envasado/sensor00/now")
        .check(status is 200))


    .exec(
      http(requestName = "Sensor03 histórico")
        .get("/api/envasado/sensor02")
        .check(status is 200))
```

La cuarta y última parte consta de la configuración de la simulación en la cual se configura el numero de usuarios para las pruebas de estrés, cabe destacar que se utilizo una inyección de usuarios tipo rampa durante 60 segundos y se configura el protocolo de la simulación HTTP.

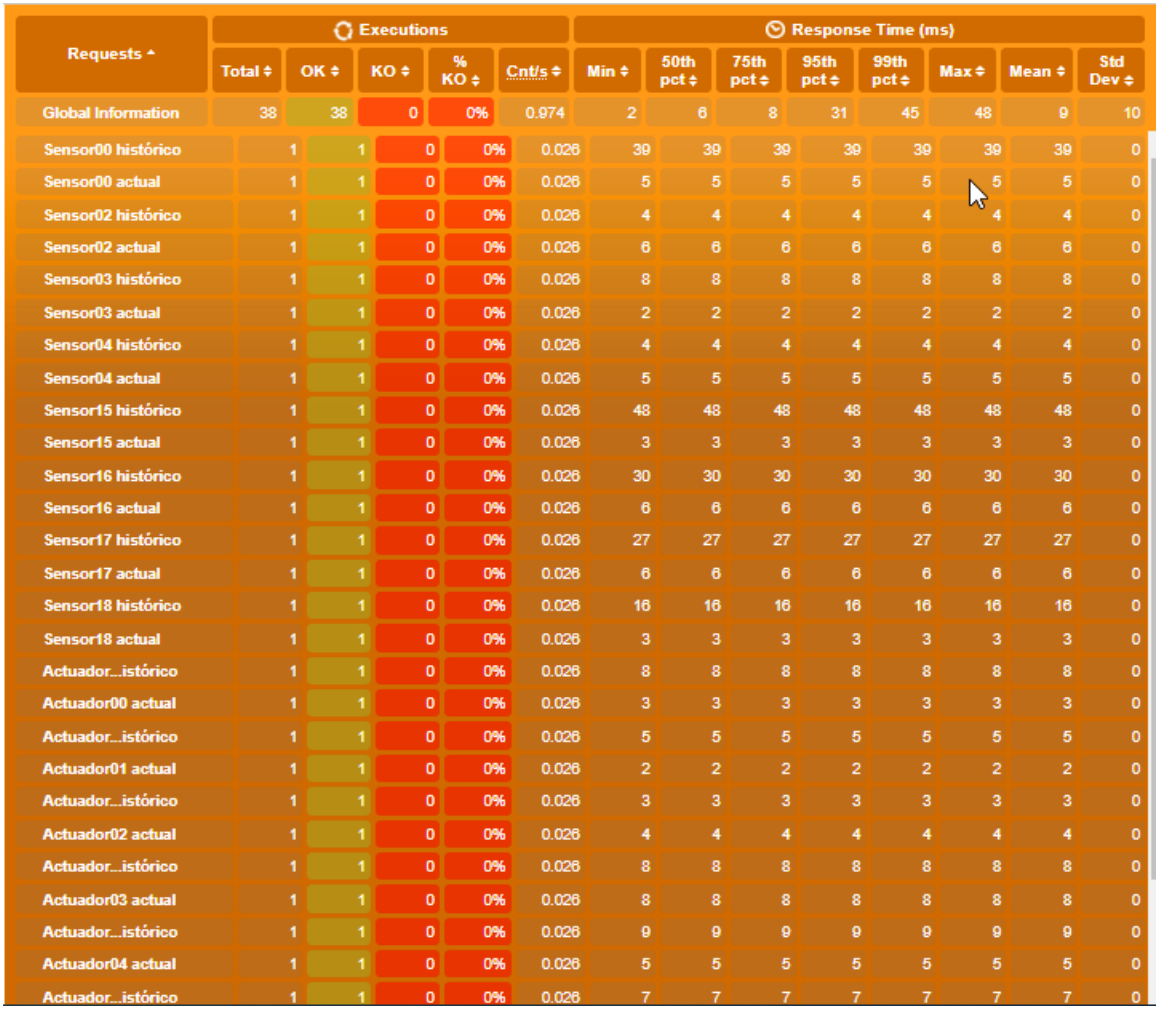
```
setUp(scn.inject(rampUsers(1500).during(60))).protocols(httpConf)
```

Ahora se realizará las pruebas de carga, en el escenario de la etapa de envasado existe un total de 38 componentes ciber físicos entre sensores y actuadores pertenecientes al caso de estudio son evaluados como se observa en la Figura 57, cada uno de los componentes tiene dos tipos de URL una URL que nos retorna todos los datos históricos o los retorna el dato actual del componente y en los dos casos su retorno es en formato JSON, para crear el escenario de Gatling se crea un script para que se realicen las peticiones.

Gatling permite acercarnos a la realidad en estas peticiones por lo cual se estresa al sistema con una entrada de usuarios tipo rampa en la cual los usuarios van ingresando uniformemente en un determinado periodo de tiempo, se lo realizó así pues es la más acorde al comportamiento que tendrían los usuarios que deseen acceder a los datos, con todos los usuarios entrando al mismo instante exactamente seria irreal en este sistema. En la primera prueba de funcionamiento se puede observar los componentes que constaron para realizar las peticiones.

Figura 57

Lista de peticiones de la prueba

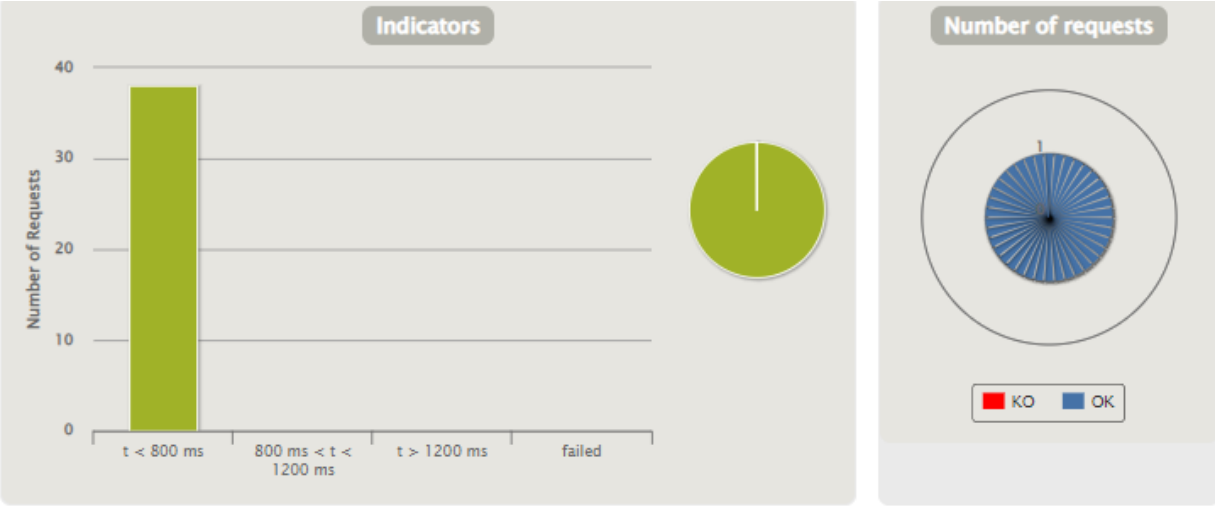


Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	38	38	0	0%	0.974	2	6	8	31	45	48	9	10
Sensor00 histórico	1	1	0	0%	0.026	39	39	39	39	39	39	39	0
Sensor00 actual	1	1	0	0%	0.026	5	5	5	5	5	5	5	0
Sensor02 histórico	1	1	0	0%	0.026	4	4	4	4	4	4	4	0
Sensor02 actual	1	1	0	0%	0.026	6	6	6	6	6	6	6	0
Sensor03 histórico	1	1	0	0%	0.026	8	8	8	8	8	8	8	0
Sensor03 actual	1	1	0	0%	0.026	2	2	2	2	2	2	2	0
Sensor04 histórico	1	1	0	0%	0.026	4	4	4	4	4	4	4	0
Sensor04 actual	1	1	0	0%	0.026	5	5	5	5	5	5	5	0
Sensor15 histórico	1	1	0	0%	0.026	48	48	48	48	48	48	48	0
Sensor15 actual	1	1	0	0%	0.026	3	3	3	3	3	3	3	0
Sensor16 histórico	1	1	0	0%	0.026	30	30	30	30	30	30	30	0
Sensor16 actual	1	1	0	0%	0.026	6	6	6	6	6	6	6	0
Sensor17 histórico	1	1	0	0%	0.026	27	27	27	27	27	27	27	0
Sensor17 actual	1	1	0	0%	0.026	6	6	6	6	6	6	6	0
Sensor18 histórico	1	1	0	0%	0.026	16	16	16	16	16	16	16	0
Sensor18 actual	1	1	0	0%	0.026	3	3	3	3	3	3	3	0
Actuador...istórico	1	1	0	0%	0.026	8	8	8	8	8	8	8	0
Actuador00 actual	1	1	0	0%	0.026	3	3	3	3	3	3	3	0
Actuador...istórico	1	1	0	0%	0.026	5	5	5	5	5	5	5	0
Actuador01 actual	1	1	0	0%	0.026	2	2	2	2	2	2	2	0
Actuador...istórico	1	1	0	0%	0.026	3	3	3	3	3	3	3	0
Actuador02 actual	1	1	0	0%	0.026	4	4	4	4	4	4	4	0
Actuador...istórico	1	1	0	0%	0.026	8	8	8	8	8	8	8	0
Actuador03 actual	1	1	0	0%	0.026	8	8	8	8	8	8	8	0
Actuador...istórico	1	1	0	0%	0.026	9	9	9	9	9	9	9	0
Actuador04 actual	1	1	0	0%	0.026	5	5	5	5	5	5	5	0
Actuador...istórico	1	1	0	0%	0.026	7	7	7	7	7	7	7	0

En la Figura 58 podemos observar que el 100% de las peticiones se completaron exitosamente y además que todas tuvieron un tiempo de respuesta menor a 800ms, esta prueba nos sirve para interpretar de buena manera la arquitectura propuesta debido a que las URL que contenían los valores históricos JSON eran de bastantes datos pues se disponía de toda la base de datos de dicho componente y aún así su funcionamiento es óptimo.

Figura 58

Prueba de peticiones con tiempo de respuesta



Pruebas de carga

De igual manera que la prueba de funcionamiento anterior el sistema debe ser estresado, pero con número variable de usuarios que realicen peticiones para validar el comportamiento del sistema ante una alta cantidad de usuarios consumiendo los diferentes servicios web que la API otorga, en un escenario que sea de igual manera que al anterior prueba basado en estrés de usuarios tipo rampa durante 60 segundos, Gatling permite configurar esto de una manera simple tan solo cambiando un parámetro para realizar las distintas pruebas se escogió diferentes cantidades de usuarios a estresar el sistema: 100, 200, 400 ,800, 1000 y finalmente 1500 usuarios.

Pruebas de carga con 100 usuarios.

En la figura 59 se puede ver la cantidad de peticiones totales que vendrían a ser 3800 debido a que 100 usuarios ingresaron a la lista de 38 URL del escenario, es importante notar además que el 100% fueron completadas con éxito y que fueron en menos de 800 ms mientras tanto en la figura 60 se puede observar el numero de peticiones por segundo y que existió picos de peticiones a lo largo de la prueba.

Figura 59

Prueba de carga con 100 usuarios

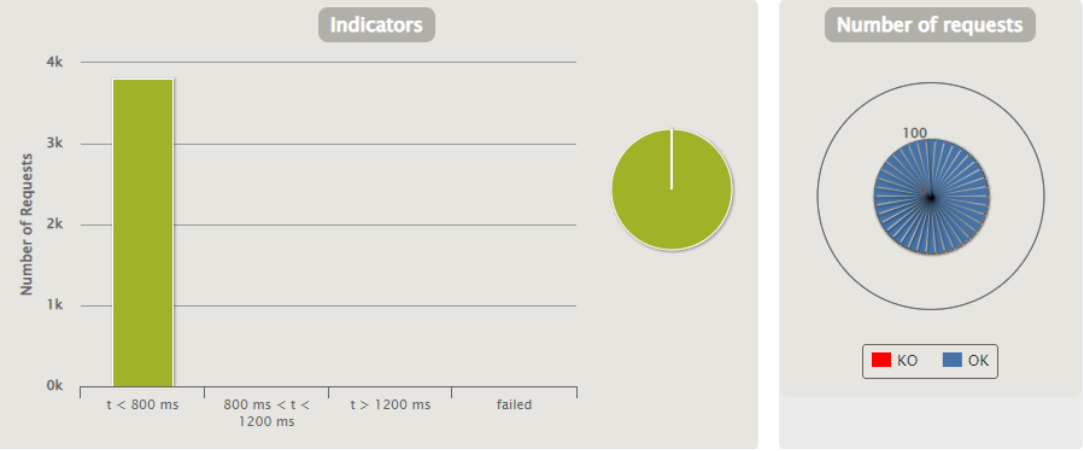
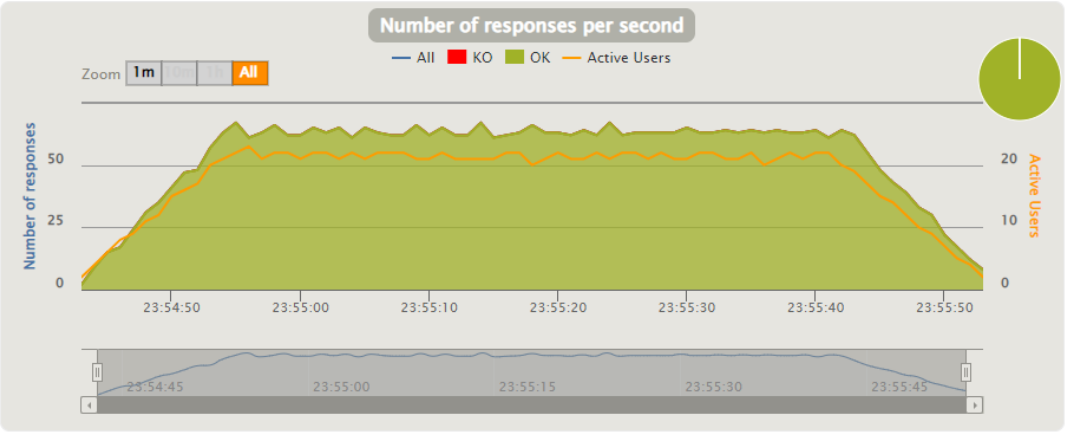


Figura 60

Pico de peticiones con 100 usuarios



Pruebas de carga con 200 usuarios.

En la figura 61 se puede ver la cantidad de peticiones en esta ocasión llegan a las 7600, el 100% de las peticiones se completaron en un tiempo menor a 800ms, en la figura 62 se puede observar también el pico de peticiones en el cual llegó a ser de 79 peticiones.

Figura 61

Prueba de carga con 200 usuarios

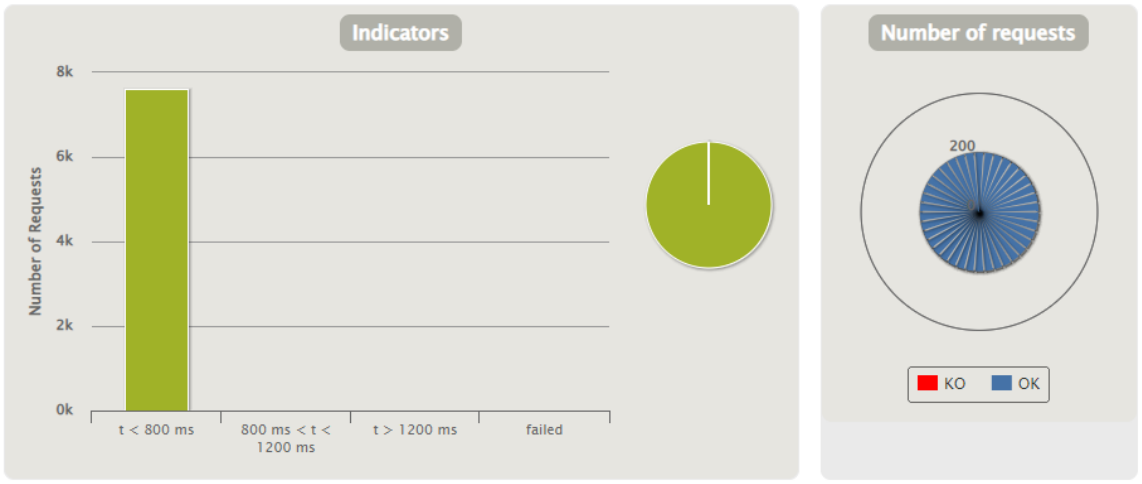
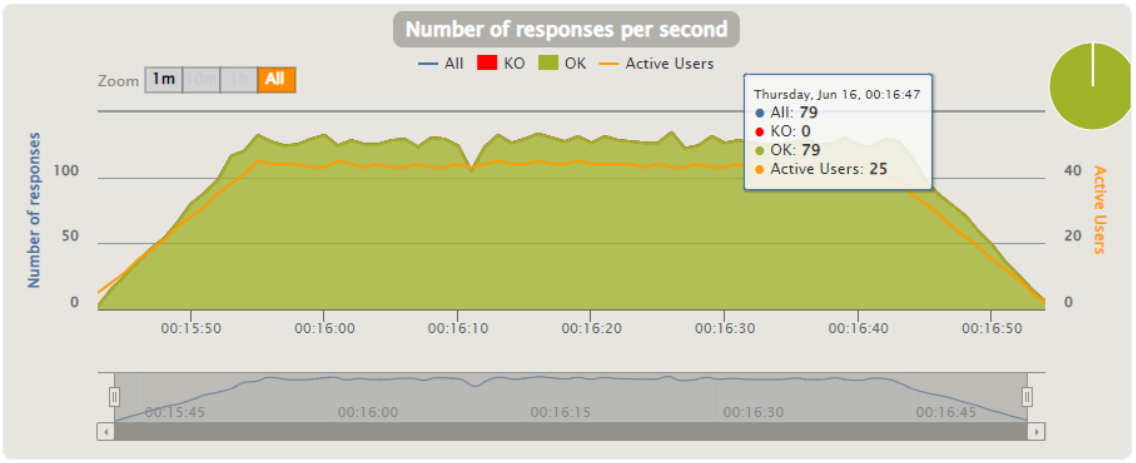


Figura 62

Pico de peticiones con 200 usuarios



Pruebas de carga con 400 usuarios.

En la figura 63 podemos ver la prueba con 400 usuarios y que llegan a hacerse 15000 peticiones y su respuesta es eficaz en menos de 800ms nuevamente y se completan el 100% de peticiones, mientras que en la figura 64 existen picos de peticiones en todo lo largo de la prueba.

Figura 63

Prueba de carga con 400 usuarios

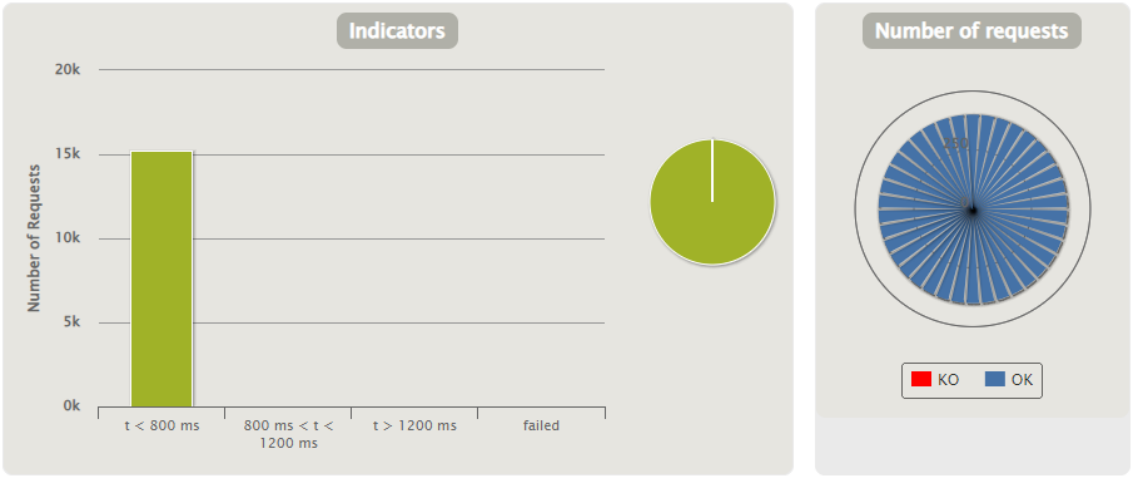
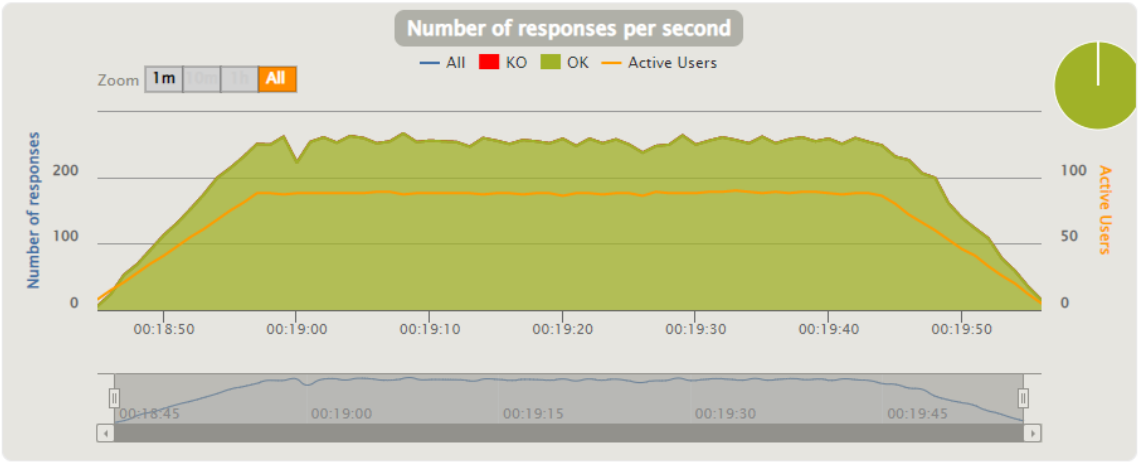


Figura 64

Pico de peticiones para 400 usuarios



Pruebas de carga con 800 usuarios.

En la figura 65 se observa que las peticiones con 800 usuarios llegan a ser 30000 y aún con este estrés el sistema lo supera en menos de 800ms y completa el 100% de las peticiones, en la figura 66 se puede observar un pico de 226 usuarios 711 peticiones cuando la prueba ya estaba por concluir y la carga no es tan uniforme.

Figura 65

Prueba de carga con 800 usuarios

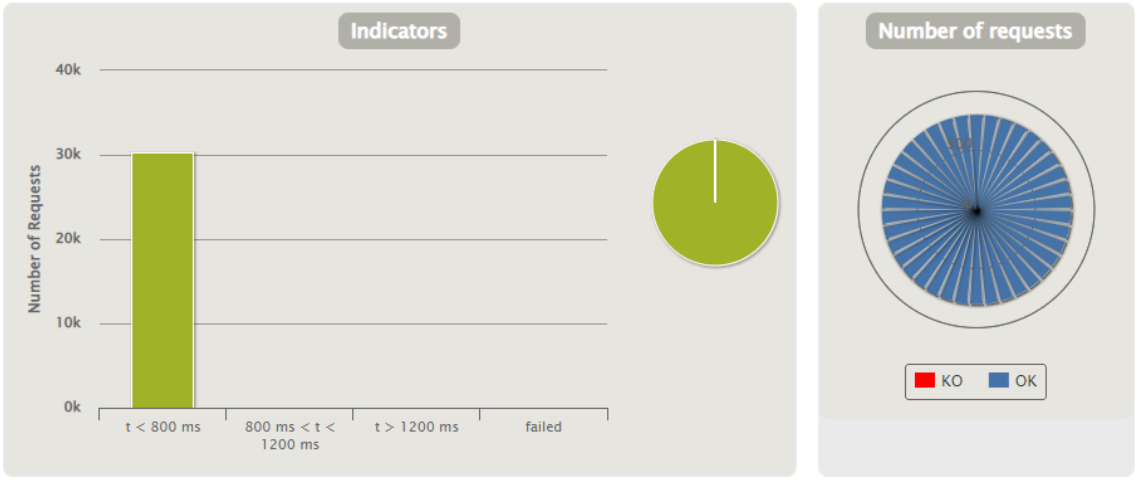
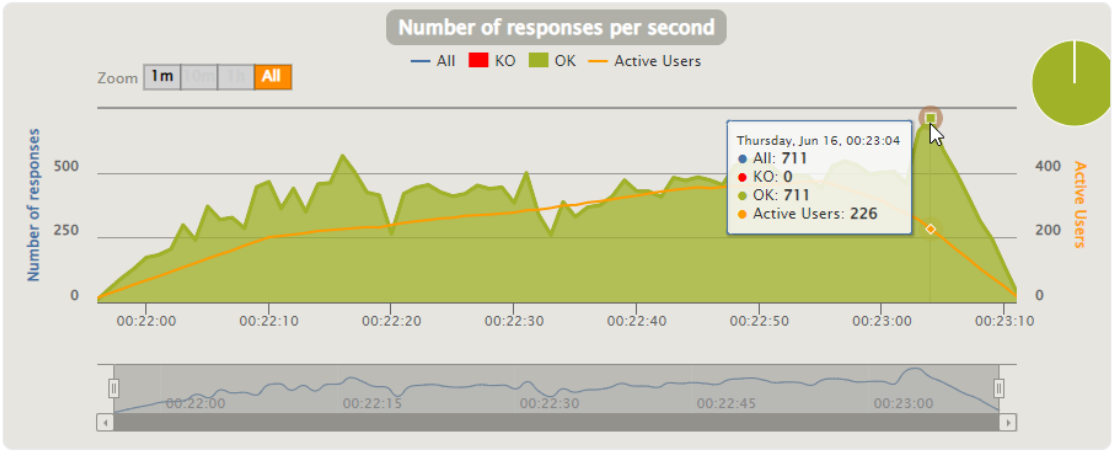


Figura 66

Pico de peticiones para 800 usuarios



Pruebas de carga con 1000 usuarios.

En la figura 67 podemos observar que son 20k de peticiones que se responden pronto es decir menor a 800ms, 7k se responden menor a 1200 ms y 10k se responden menor a 1200ms aquí el sistema ya tiene una carga bastante elevada y los datos de los 10k después de los 1200ms ya no servirían para realizar gráficas en tiempo real, en la figura 68 existe un pico de 1222 peticiones al final de la prueba con 447 usuarios.

Figura 67

Prueba de carga con 1000 usuarios

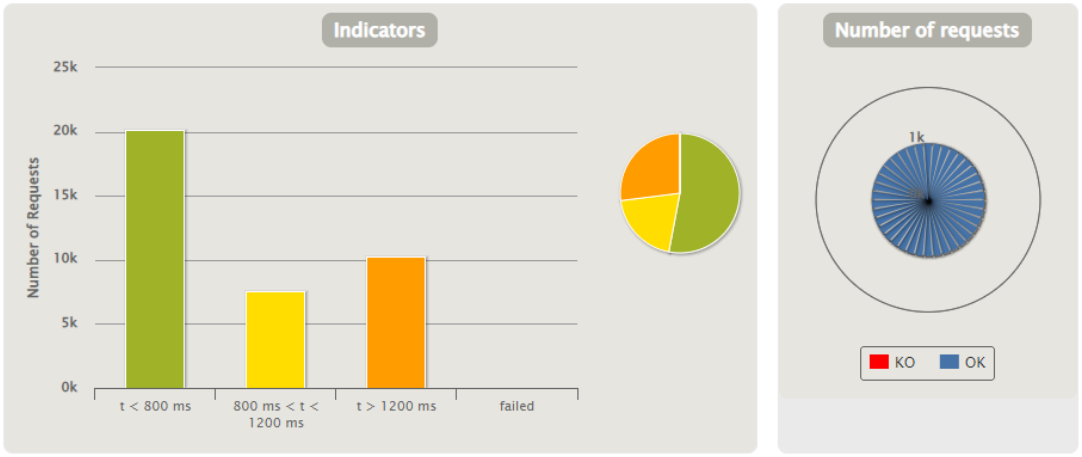
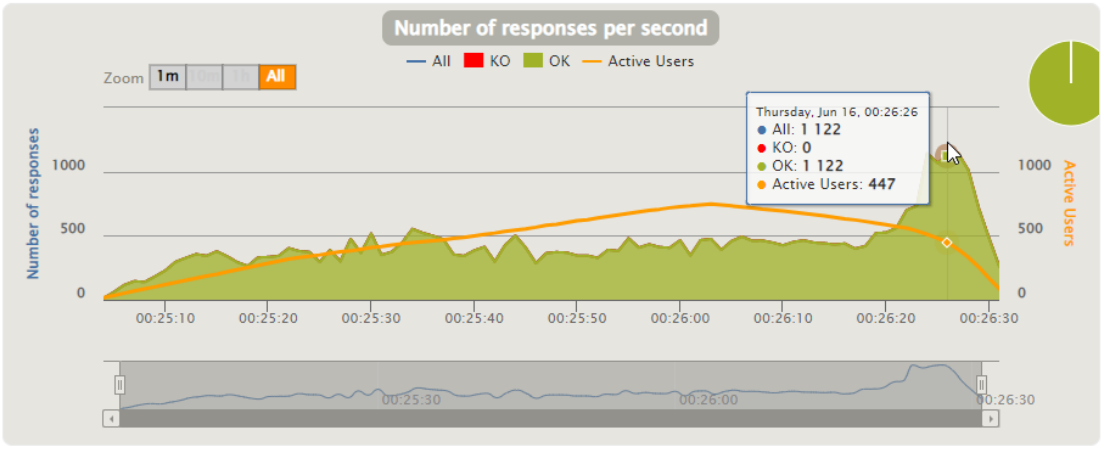


Figura 68

Pico de peticiones para 1000 usuarios



Pruebas de carga con 1500 usuarios.

La última prueba se realizó con 1500 usuarios de los cuales 30k peticiones fueron en un tiempo de respuesta de 1200 segundos, 18k, en menos de 800ms y 8k entre 800 y 1200 ms, con 1500 usuarios, finalmente se concluye que sería imposible crear gráficos históricos con 1500 usuarios estresando el sistema, en la figura 70 se puede ver que los picos máximos se dieron al final de la prueba con 1131 peticiones con 847 usuarios.

Figura 69

Prueba de carga con 1500 usuarios

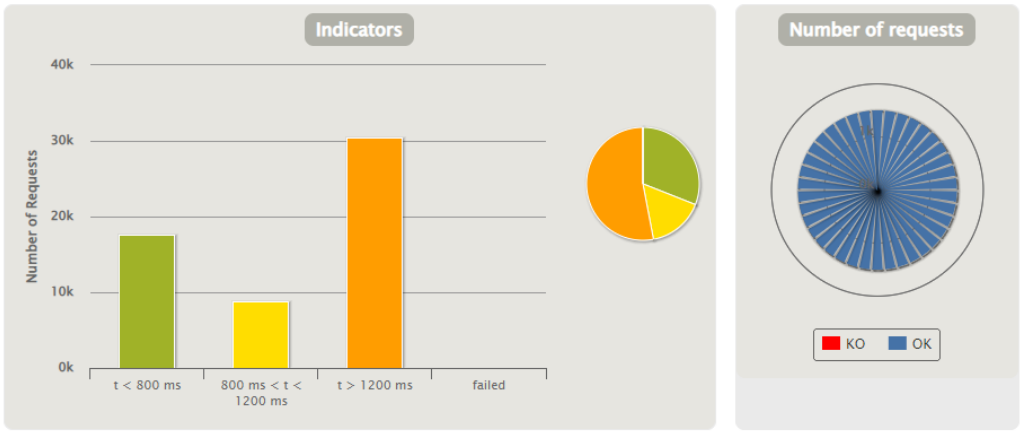
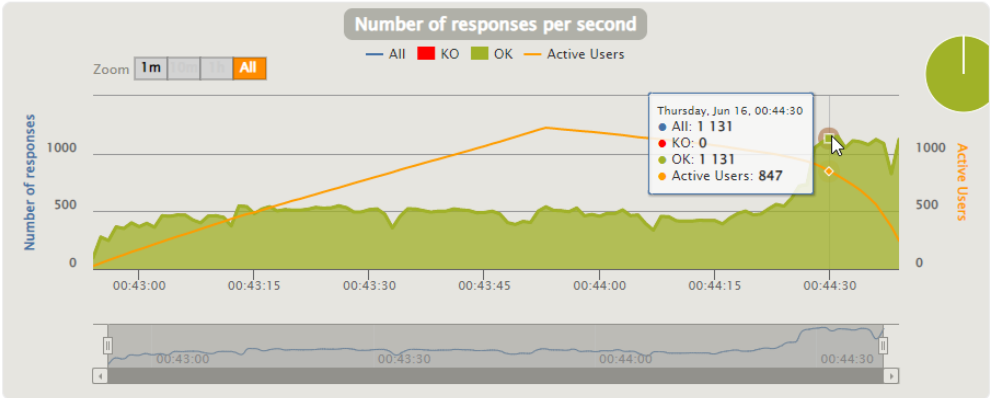


Figura 70

Pico de peticiones para 1500 usuarios



En la tabla 10 finalmente se recopila los datos de todas las pruebas de carga realizadas, se puede observar que hasta 800 usuarios el sistema funciona perfectamente hasta con 507 peticiones por segundo todas las respuestas son menores a 800ms, en un sistema que entregue datos de sensores en tiempo real esto es lo que se busca que el tiempo de respuesta sea el menor posible, mientras que para 1000 usuarios y 1500 usuarios el sistema no colapsa pero entrega las pruebas a destiempo por lo cual se demoraría demasiado en por ejemplo activar una alarma que sea de urgencia pues sería alrededor 1 segundo que debería tomar una acción como cerrar válvulas o retraer cilindros.

Tabla 10

Datos recopilados totales de pruebas

Usuarios	Peticiones Totales	Máximas Pets/seg	%Éxito	%Respuesta t<800ms	%Respuesta t<1200ms
100	3800	63	100%	100%	0%
200	7600	127	100%	100%	0%
400	15200	254	100%	100%	0%
800	30400	507	100%	100%	0%
1000	38000	633	100%	82%	18%
1500	57000	950	100%	32%	68%

Pruebas de usabilidad

Para evaluar la experiencia del usuario frente a la arquitectura propuesta se realizó una evaluación utilizando el sistema de escalas de usabilidad (SUS), evaluación mediante la cual se realizó a 10 personas que interactuaron con la capa de aplicación del caso de estudio la cual es la etapa de envasado.

Las 10 personas que fueron evaluadas fueron estudiantes y graduados de la carrera de Ingeniería Electrónica, los cuales dieron opiniones positivas sobre que tan amigable para el

usuario es el sistema. El sistema de usabilidad consta de 10 preguntas y cada pregunta puede ser evaluada con números del 1 al 5, tanto que 1 es totalmente en desacuerdo y 5 totalmente de acuerdo.

Como se observa en la tabla 11 el promedio del puntaje SUS fue de 88.5/100 con lo cual se comprueba que el sistema es amigable y cumple con ser amigable para el usuario que desee implementar el sistema. Además, se pudo ver que el sistema llamó a la atención a los encuestados pues la Industria 4.0 es una tecnología que esta emergente y aún está en desarrollo por lo cual es un tema que despierta críticas y opiniones positivas.

Tabla 11

Datos obtenidos de pruebas de usabilidad

Preguntas / Usuarios	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**Puntaje SUS**
U1	4	1	5	1	5	1	3	1	4	1	90
U2	5	2	5	1	5	2	5	1	3	2	87,5
U3	3	1	5	3	5	1	4	1	4	1	85
U4	5	1	4	2	4	1	3	1	5	1	87,5
U5	4	1	5	2	5	1	4	1	4	1	90
U6	5	1	4	3	4	1	5	1	4	1	87,5
U7	3	1	3	1	3	1	5	1	4	1	82,5
U8	4	1	5	3	5	1	4	2	5	2	85
U9	5	1	5	1	4	1	4	1	5	1	95
U10	5	1	5	1	4	1	5	1	5	2	95
										Promedio	88,5

Capítulo 6: Conclusiones y Recomendaciones

Conclusiones

Se realizó un trabajo investigativo de las tecnologías, herramientas, arquitecturas que permitió el diseño y la implementación de la arquitectura basada en la web of things para fábricas inteligente en las que los componentes ciber físicos de la fábrica son vistos como componentes web y utilizando una arquitectura orientada a servicios web se logró implementar una arquitectura de software fiable y en el cual su control industrial se encuentra en la nube y es realizado por un orquestador de microservicios.

Mediante la implementación de la arquitectura en su totalidad con RESTful, Web Of Things y Servicios Web, se pudo estandarizar la comunicación entre sistemas independiente del protocolo de comunicación utilizado por los diferentes sistemas ciberfísicos. Además, se da la posibilidad a las fábricas antiguas que utilicen PLC’s sin salida a la nube, puedan mediante un Gateway o Pasarela realizar peticiones basadas en el protocolo HTTP para enviar y recibir datos en formato JSON y hacer uso de la arquitectura propuesta.

Se comprobó que mediante la utilización de servicios web para la emulación de componentes ciber físicos y el análisis de sus mecanismos se puede simular componentes reales con una aproximación muy cercana a la realidad, capaz de integrarse para formar parte de un escenario de prueba el cual fue la etapa de envasado para obtener pruebas de validación de la arquitectura propuesta.

Mediante una arquitectura orientada a servicios y la coordinación de estos microservicios es posible realizar las secuencias de control antes ejecutadas de forma local, ahora en la nube con la interacción de todos los componentes mediante una orquestación y que pueda integrarse con servidores remotos.

Mediante Ignition se pudo implementar de manera satisfactoria la capa de aplicación en la cual podemos obtener el monitoreo en tiempo real y control de los diferentes tipos de

producción que la fábrica ofrece, así como ingresar ordenes de producción y editar o monitorear la Thing Description de los componentes ciber físicos, así como interactuar con el proceso en su etapa de producción.

Mediante un escenario simulado se logró verificar el correcto funcionamiento de la arquitectura desplegada, en su análisis se pudo observar que cumplió todos los lineamientos de los requisitos de diseño interpuestos como la generación de alarmas, reportes de producción, monitoreo en tiempo real, modos de funcionamiento, control a la producción y producción flexible.

Se realizó pruebas de estrés al sistema para validar su correcto funcionamiento, así como concluir que el sistema a una inyección de 1000 usuarios sus respuestas tienen un tiempo de respuesta de alrededor de 1200 ms siendo un tiempo no aceptable para un monitoreo en tiempo real. Las pruebas inferiores a ese número de usuarios resultaron exitosas aun existiendo picos temporales de usuarios en medio de la prueba.

Con las pruebas de usabilidad se obtuvo un puntaje bastante alto en la escala SUS el cual fue de 88,5 sobre 100, por lo cual se puede concluir que el sistema es amigable para el usuario y que permite a un operario por ejemplo que pasa las 24 horas del día monitoreando variables que la interfaz sea cómoda, intuitiva y completa.

Recomendaciones

Para la utilización de la arquitectura es fundamental que esta se encuentre en un servidor virtual con internet para hacer uso de las prestaciones como el email la generación de alarmas y reportes de producción, monitoreo remoto. También podría funcionar en un servidor local con Mosquitto, MYSQL y el software de aplicación montado en el mismo cuando no se desee tener salida a la nube.

En la presente arquitectura se utilizó Ignition que es un software para crear sistemas SCADA, pero es de pago, por lo cual se recomienda si se hace uso de este estar pendiente de

cuando terminen las dos horas de prueba para recargar el sistema y evitar inconvenientes como ignorar alarmas o reportes que el sistema genere.

Para el diseño de los componentes ciber físicos es importante ver a los componentes a simular como lo más cercano a la realidad incluyendo sus tiempos de arranque, sensores incluidos, etc. Además, deben recibir y generar información mediante porciones de código que pueden estar validadas con condiciones propias del escenario como el nivel aumente solo si el actuador de válvula esté abierto.

Trabajos Futuros

Para trabajos futuros se propone.

- Crear la capa de aplicación con tecnologías como CCS, REACT que permitan visualizar los datos de forma interactiva al igual que lo realiza Ignition, generación de alarmas, botones y prestaciones que este software ofrece.
- Implementar protocolos de seguridad para los datos sensibles de la fábrica pues al estar en la nube se encuentran expuestos, pero se tiene el beneficio que si se cae un microservicio no desencadena nada más que su caída.
- Implementar la arquitectura físicamente y en otro escenario para validar el correcto funcionamiento, también será parte del trabajo realizar cambios debido que al pasar de simulación a implementación existen cosas que se deben tener en cuenta.

Acrónimos

- CPS. Sistemas Ciberfísicos
- CPPS. Sistemas Ciberfísicos de producción
- HTTP. Hypertext Transfer Protocol
- IoT. Internet de las Cosas
- IT. Tecnologías de la Información
- MQTT. Messege Queuing Telemetry Transport (Transporte de telemetría)
- OT. Tecnologías de Operación
- REST. representational state transfer (Transferencia de estado representacional)
- SLR. Revisión Sistemática de la Literatura
- SMS. Mapeo Sistemático de la Literatura.
- SOA. Arquitectura Orientada a Servicios
- WoT. Web de las Cosas
- JS: JavaScript.

Referencias Bibliográficas

Kaschel, H., & Pinto, E. (2020). *Análisis del estado del arte de los buses de campo aplicados al control de procesos industriales.* Universidad de Santiago de Chile, Santiago de Chile.

Binfeng, L., Jianguo, C., Changyu, L., & Yilong, L. (2017). Design and Implementation of a Lightweight Electronic Village System Based on REST Web Service. *International Journal of Future Generation Communication and Networking*. doi:http://dx.doi.org/10.14257/ijfgcn.2017.10.1.14

Biswajeeban, M., & Attila , K. (2021). Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements. *Energies 2021*. doi:https://doi.org/10.3390/en14185817

Cano, M. (2018). *Industria 4.0, Excelencia Operativa.* Obtenido de https://manuelguerrerocano.com/it_ot_convergencia/

Celaya, I., Ramírez, M., Naval, C., & Arbués, E. (2020). Usos del podcast para fines educativos. Mapeo sistemático de la litertura en WoS y Scopus. *Revista Latina de Comunicación Social*. Obtenido de https://doi.org/10.4185/RLCS-2020-1454

Chaniotis, I., Kyriakou , K.-I., & Tselikas , N. (2015). Is Node.js a viable option for building modern web applications? A performance evaluation study. *Computing*. doi:https://doi.org/10.1007/s00607-014-0394-9

Chi, C., Dang, Q., & Vu, H. (2022). Low-power Mesh Network Based on Message Queue Telemetry Transport Broker for Industrial IoT. *Sensors and Materials*.

Cortés, C., Landeta, J., & Chacón, J. (s.f.). El Entorno de la Industria 4.0: Implicaciones y Perspectivas futuras. *Tecnológica*.

García Peñalvo, F. J. (2017). *Revisión sistemática de literatura en los Trabajos de Final de Máster y en las Tesis Doctorales.* Universidad de Salamanca. Obtenido de https://repositorio.grial.eu/bitstream/grial/813/1/SLR-TFM-Tesis.pdf

Garcia, F. (2020). *Revisión de las tecnologías presentes en la industria 4.0.* Universidad Industrial de Santander, Colombia. Obtenido de https://www.redalyc.org/journal/5537/553768132019/553768132019.pdf

Gokhale, P., Bhat, O., & Bhat, S. (2018). Introduction to IOT. *IARJSET*. doi:10.17148/IARJSET.2018.517

Guinard , D., & Trifa, V. (2016). *Building the Web of Things.* Manning.

Hans, K. (2015). *Mass Customization.* Apress L. P.

Haro, E., Guarda, T., Alex, P., & Quiña, G. (2019). Desarrollo backend para aplicaciones web, Servicios Web Restful: Node.js vs Spring Boot. *Revista Ibérica de Sistemas e Tecnologias de Informação*.

Hozdic, E., & Butala, P. (2020). Concept of Socio-Cyber-Physical Work Systems for Industry 4.0. *Hrack*. doi:https://doi.org/10.17559/TV-20170803142215

Ignition. (2021). *Ignition*. Obtenido de https://docs.chariot.io/display/CLD80/MQTT+Engine

ISOTools. (2018). *ISOTools.* Obtenido de https://www.isotools.org/2018/07/12/industria-4-0-que-debemos-saber/

Jácome, C. (2019). *Actuadores Neumáticos.* Cali.

Jacopo, S., Tamburri, D., & Den Heuvelb, W.-J. (2018). The pains and gains of microservices: A Systematic grey literature review. *Sciencedirect*. doi:https://doi.org/10.1016/j.jss.2018.09.082

Kinsta. (2021). *Kinsta.* Obtenido de https://kinsta.com/es/base-de-conocimiento/que-es-node-js/

Kitchenham, B., Budgen, D., & Brereton, O. (2011). *Using mapping studies as the basis for further research – A participant-observer case study.* doi:https://doi.org/10.1016/j.infsof.2010.12.011.

Kroll, J., Richardson, I., Prikladnicki, R., & Audy, J. (2018). Systematic literature reviews in software engineering–a tertiary study. Information and software technology. doi:https://doi.org/10.1016/j.infsof.2017.08.011

Kubr, J., Novikov, K., Horejsi, P., Kleinova, J., & Krakora, D. (2021). Connecting a virtual production and PLC. *MM SCIENCE JOURNAL*.

Lequerica, J. (2003). *Web Services.*

Mabkhot, M., Abdulrahman, A.-A., Salah, B., & Alkhalefah, H. (2018). Requirements of the Smart Factory System: A Survey and Perspective. *Machines*. doi:https://doi.org/10.3390/machines6020023

Mahedero, F. (2020). *Desarollo de una aplicación IoT para el envío de imágenes mediante el protocolo MQTT.* Universidad Politécnica de Valencia, Valencia.

Martins, J. A., Mazayev, A., & Correia, N. (2017). Hypermedia APIs for the Web of Things. *IEEE Access*. doi:10.1109/ACCESS.2017.2755259

Mazzara, M., Dragoni, N., Bucchiarone, A., Giaretta, A., & Lars, S. (2020). Microservices: Migration of a Mission Critical System. *IEEE*. doi:10.1109/TSC.2018.2889087.

mitsubishielectric. (2022). *mitsubishielectric.* Obtenido de https://mx.mitsubishielectric.com/fa/es/solutions/efactory/enabling-technologies/mes-erp-and-cloud-appliances/iot-gateway/iot-gateway

Moguel, J. (2018). *Una arquitectura orientada a servicios y dirigida por eventos para el control inteligente de UAVs multipropósito.* Universidad de Extramadura.

Napoleone, A., Macchi, M., & Pozzetti, A. (2020). A review on the characteristics of cyber-physical systems for the future smart factories. *ScienceDirect*. doi:https://doi.org/10.1016/j.jmsy.2020.01.007.

Osterrieder, P., Budde, L., & Friedli, T. (2020). The smart factory as a key construct of industry 4.0: A systematic literature review. *Elsevier*. doi:https://doi.org/10.1016/j.ijpe.2019.08.011

Pérez, L. (2018). Las tecnologías del cambio IT-OT. *AADECa*. Obtenido de https://www.editores.com.ar/sites/default/files/aa9_perez_tecnologias.pdf

Saleem, M. (2017). Security Problems of SOA Applications. *International Journal Of Security and ITS applications*.

Science, W. O. (2021). *Web Of Science*. Obtenido de https://www.webofscience.com/

Somayya, M., Siddharth, T., & Ramaswam, R. (2015). Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications*. doi:10.4236/jcc.2015.35021.

Sudhanshu, M., & Mukherjee, K. (2019). An Energy Efficient Architecture of IoT Based on Service Oriented Architecture (SOA). *INFORMATICA-JOURNAL OF COMPUTING AND INFORMATICS*.

Viktor , M.-S. (2013). *Big data : la revolución de los datos masivos.* Madrid: Turner, 2013.

Villada Romero, J. L. (2015). *Instalación y configuración del software de servidores web.* Málaga: IC Editorial.

Waltman, L., & Van Eck, N. (2013). *VOSviewer Manual.* Univeristeit Leiden. Obtenido de https://www.vosviewer.com/documentation/Manual_VOSviewer_1.6.17.pdf

Apéndices