



Diseño e implementación de un sistema domótico basado en la web of things (WoT) para la orquestación y descubrimiento de componentes cyber físicos.

García Silva, Leandro Josué

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.

24 de febrero del 2022



Tesis_Garcia_Josue.pdf

Scanned on: 14:32 February 24, 2022 UTC



DARWIN OMAR
ALULEMA
FLORES



Overall Similarity Score



Results Found



Total Words in Text

Identical Words	559
Words with Minor Changes	188
Paraphrased Words	1541
Omitted Words	0



**DEPARTAMENTO DE ELECTRICA, ELECTRÓNICA Y
TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA,
AUTOMATIZACIÓN Y CONTROL**

CERTIFICACIÓN

Certifico que el trabajo de titulación, “**Diseño e implementación de un sistema domótico basado en la web of things (WoT) para la orquestación y descubrimiento de componentes cyber fisicos**” fue realizado por el señor **García Silva, Leandro Josué**, el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 22 de febrero del 2022

Firma:



.....
Dr. Alulema Flores, Darwin Omar

C. C: 1002493334



**DEPARTAMENTO DE ELECTRICA, ELECTRÓNICA Y
TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA,
AUTOMATIZACIÓN Y CONTROL**

RESPONSABILIDAD DE AUTORÍA

Yo, **García Silva, Leandro Josué** con cédula de ciudadanía N°1804608154, declaro que el contenido, ideas y criterios del trabajo de titulación: **"Diseño e implementación de un sistema domótico basado en la web of things (WoT) para la orquestación y descubrimiento de componentes cyber físicos"**, 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, 22 de febrero del 2022

Firma:

.....
García Silva, Leandro Josué

C.C.: 1804608154



**DEPARTAMENTO DE ELECTRICA, ELECTRÓNICA Y
TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA,
AUTOMATIZACIÓN Y CONTROL**

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **García Silva, Leandro Josué** con cédula de ciudadanía N°1804608154, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**Diseño e implementación de un sistema domótico basado en la web of things (WoT) para la orquestación y descubrimiento de componentes cyber físicos**", en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 22 de febrero del 2022

Firma:

.....
García Silva, Leandro Josué

C.C.: 1804608154

Dedicatoria

Este trabajo se lo dedico a mi madre Elsa, por su cariño tan sincero, su paciencia y sabiduría, por ser mi apoyo incondicional y por ser mi padre y madre a la vez.

A mi abuelita Fanny, quien siempre me recibe con una sonrisa, me consciente y me guía.

A mi hermano Mateo, quien es mi compañía, apoyo y compañero de todo.

A mis amigos, quienes siempre han creído en mi y se han convertido en parte de mi familia.

García Silva Leandro Josué.

Agradecimiento

Un camino lleno de pruebas, dificultades y obstáculos, pero sin duda alguna también lleno de experiencias, alegrías, anécdotas y conocimiento. Así resultó mi carrera de ingeniería en electrónica, automatización y control. Siguiendo a mi corazón avancé en este recorrido, largo pero apasionante, complicado pero reconfortante; unas veces se ganan y otras se pierden, no obstante, en este sendero he ganado más de lo que perdido, he ganado amigos, historias, amores, madurez, fortaleza y carácter para seguir adelante en la búsqueda de mis sueños.

En primer lugar, quiero darle gracias infinitas a mi madre Elsa Noemí y mi abuelita Fanny, gracias por tanto amor y comprensión, por no dejarme caer en ningún momento y permanecer a mi lado dándome el apoyo más lindo, el amor de una madre, gracias por enseñarme que ninguna adversidad es imposible de vencer.

Gracias a mi hermano Mateo, quien a pesar de ser unos años menor ha ayudado a forjar mi carácter y me ha brindado la mano en los momentos más difíciles, a mi tío Hugo quien ha sabido aconsejarme siempre y a mi padre por la ayuda económica y educativa.

Gracias a todos mis amigos, en quienes pude refugiarme en tiempos difíciles y levantaron mis ánimos, gracias por todo su apoyo y cariño, por ser mi familia en otra ciudad, gracias a mis amigos Josafat y Nikolas, quienes siempre están prestos para una conversación, una salida y un consejo, gracias por compartir tiempo conmigo los domingos, por darme ánimos y su apoyo tan sincero.

Finalmente, un agradecimiento especial para mi tutor Ing. Darwin Alulema, quien supo guiarme de gran manera con sus conocimientos, experiencia y enseñanzas en la exitosa consecución del presente proyecto.

García Silva Leandro Josué.

Índice General

8

Similitud de contenido con herramienta anti plagio	2
Certificación del trabajo de titulación.....	3
Responsabilidad de autoría	4
Autorización de publicación	5
Dedicatoria.....	6
Agradecimiento.....	7
Índice General	8
Índice de Tablas	16
Índice de Figuras	17
Resumen	21
Abstract	22
Capítulo I	23
Marco Metodológico.....	23
Antecedentes	23
Justificación e Importancia	27
Alcance del Proyecto.....	30
Objetivos	33
General.....	33
Específicos	33
Estado del Arte	34

	9
Síntesis de evidencia disponible.....	36
Mapeo Sistemático de la Literatura.....	36
Objetivos.....	37
Método de investigación.....	39
Definir preguntas para la investigación.....	40
Establecer los criterios de averiguación primarios y secundarios.....	41
Definir las cadenas de búsqueda.....	41
Establecer juicios de inclusión y exclusión.....	43
Determinar la estrategia de extracción de datos.....	44
Clasificación de los artículos.....	44
Resultados.....	46
Conclusiones del SMS.....	47
Revisión Sistemática de la Literatura.....	52
Objetivos.....	52
Método de investigación.....	53
Definir las preguntas de investigación.....	54
Establecer los criterios de búsqueda primarios y secundarios.....	56
Definir cadena de búsqueda.....	56
Fijar los criterios de inclusión y exclusión.....	57
Determinar la estrategia de extracción de los datos.....	58
Clasificación de los documentos extraídos.....	58

Resultados	10
Conclusiones del SLR	60
Definición de las características del sistema	65
Sistema domótico	66
Microservicios	67
Sistema ciber físico.....	68
Control y Automatización	68
Capítulo II	70
Tecnologías Relacionadas.....	70
Domótica	70
Características de la Domótica	70
Ventajas de la Domótica	71
Desventajas de la Domótica	71
Inmótica.....	72
Características de la Inmótica.....	72
Ventajas de la Inmótica.....	73
Desventajas de la Inmótica	74
Arquitecturas de servicios	75
SOA (Service Oriented Architecture)	75
Características de SOA.....	75
Ventajas de SOA.....	76

Desventajas de SOA	11
Desventajas de SOA	77
Arquitectura de Microservicios	78
Características de una arquitectura de microservicios.....	79
Ventajas de una arquitectura de microservicios.....	79
Desventajas de una arquitectura de microservicios.....	80
REST (Representational State Transfer).....	80
Características de REST	81
Ventajas de REST.....	82
Desventajas de REST	82
Tecnologías de Back-End	83
JAVA.....	84
Processing.....	85
MQTT	85
Tecnologías de Front-End	86
HTML.....	87
CSS	87
JavaScript.....	88
Agente de orquestación y descubrimiento	88
Orquestación y descubrimiento de servicios	88
Seguridad	90
Herramientas de Software	91

SpringBoot.....	12
Hashicorp Consul	91
Cónsul Agent.....	93
Cónsul Agent.....	94
Sistemas de ingeniería, informáticos y de comunicación.....	95
Sistemas Ciber-físicos (CPS, Cyberphysical Systems)	96
Componentes domóticos.....	96
Sensores y actuadores	96
Plataformas de Hardware.....	97
Arduino	97
Herramientas de pruebas	99
Gatling	99
Sistema de escalas de usabilidad	100
Capítulo III	102
Diseño de la Arquitectura.....	102
Requisitos de diseño	102
Estructura de diseño del sistema.....	102
Frontend.....	106
Estructura de la aplicación de verificación	106
Funcionamiento de los componentes de la aplicación de verificación.....	107
Controles de los componentes de la aplicación.....	108
Componente de las alertas de la aplicación de verificación.....	109

	13
Backend	109
Estructura general del backend.....	110
Funcionamiento general del backend.....	111
Estructura interna de los microservicios	111
Servicio de Gateway y Proxy.....	112
Microservicios de los componentes ciberfísicos.	112
Agente orquestador y de descubrimiento.	115
Estructura del agente orquestador y de descubrimiento.....	115
Funcionamiento del agente orquestador y de descubrimiento.	116
Casa domótica	116
Estructura interna de la casa domótica	117
Funcionamiento de la casa domótica	117
Capítulo IV	119
Implementación de la Arquitectura.....	119
Estructura general de la arquitectura implementada.....	119
Arquitectura final.	120
Descripción de la casa domótica.....	123
Estructura externa de la casa domótica	123
Sensores.....	124
Actuadores.	124
Estructura interna de la casa domótica	125

	14
Especificaciones técnicas de los componentes de la casa domótica.....	126
Conexiones de los componentes de la casa domótica.	127
Funcionamiento de la casa domótica.....	130
Software.....	132
Programación de la casa domótica.....	132
Conexión con Arduino.	133
Estructura y funcionamiento del programa de la casa domótica.	134
Frontend – Aplicación de Verificación.....	136
Configuraciones de la aplicación de verificación.....	137
Backend – Arquitectura del servidor.....	141
Estructura de los microservicios.	142
Configuración de las clases del microservicio.	144
Microservicios extra.....	154
Estructura de las bases de datos de los microservicios.....	155
Agente orquestador y de descubrimiento.	156
Broker MQTT.	159
Capítulo V.....	161
Pruebas de Validación.....	161
Pruebas de funcionamiento.....	161
Prueba de carga con 100 usuarios.	169
Prueba de carga con 200 usuarios.....	171

	15
Prueba de carga con 400 usuarios	172
Prueba de carga con 800 usuarios	173
Prueba de carga con 1000 usuarios	174
Prueba de carga con 5000 usuarios	175
Pruebas de usabilidad	176
Capítulo VI.....	179
Conclusiones y Recomendaciones	179
Conclusiones.....	179
Recomendaciones.....	181
Trabajos Futuros.	182
Acrónimos.....	183
Referencias.....	184
Anexos.....	192

Índice de Tablas

Tabla 1 Trabajos relacionados	26
Tabla 2 Preguntas de investigación para el desarrollo del SMS	40
Tabla 3 Términos utilizados en las cadenas de búsqueda del SMS.	42
Tabla 4 Preguntas del proceso de investigación del SLR	55
Tabla 5 Términos definidos para la cadena de búsqueda del SLR.	56
Tabla 6 Sensores y actuadores.....	97
Tabla 7 Características del ESP8266 – NodeMCU V.3.	98
Tabla 8 Requisitos funcionales.....	103
Tabla 9 Requisitos no funcionales.....	105
Tabla 10 Especificaciones técnicas de los componentes de la casa domótica.	126
Tabla 11 Componentes electrónicos con sus pines y su tipo de comunicación.	130
Tabla 12 Funcionamiento de la casa domótica.....	131
Tabla 13 Actuador, acción y tópico del broker.	136
Tabla 14 Botones de la aplicación de verificación y sus atributos.....	140
Tabla 15 Resumen de las pruebas de carga	176
Tabla 16 Resultados del test de usabilidad.	177

Índice de Figuras

Figura 1 Esquema general del proyecto.	33
Figura 2 Densidad de búsquedas por palabras.....	35
Figura 3 Densidad de búsquedas por países.....	35
Figura 4 Proceso de investigación del Estado del Arte.....	36
Figura 5 Proceso de elaboración del SMS.....	40
Figura 6 Cantidad total de artículos en cada fase del SMS.....	46
Figura 7 Proceso de desarrollo para el SLR.....	54
Figura 8 Cantidad de artículos extraídos en todo el proceso del SLR.....	60
Figura 9 SOA en un modelo de negocio.....	76
Figura 10 Diagrama general de una arquitectura de microservicios	78
Figura 11 Spring Boot IDE.....	92
Figura 12 Navegador redireccionado al localhost.....	93
Figura 13 Sistema basado en Consul.....	95
Figura 14 ESP8266 – NodeMCU V.3.....	98
Figura 15 Estructura general del diseño de la arquitectura.....	103
Figura 16 Bosquejo de la aplicación de verificación.....	107
Figura 17 Funcionamiento de los componentes de la aplicación de verificación.....	108
Figura 18 Ejemplo de controles de un componente.....	108
Figura 19 Ejemplo de alerta emergente.....	109
Figura 20 Estructura general del backend	110
Figura 21 Diagrama de flujo del funcionamiento general del backend	111
Figura 22 Estructura del servicio de Gateway y Proxy.....	112
Figura 23 Estructura de un microservicio.....	114
Figura 24 Funcionamiento de un microservicio.....	114

	18
Figura 25 Estructura del agente orquestador y de descubrimiento.	115
Figura 26 Funcionamiento del agente orquestador y de descubrimiento.	116
Figura 27 Estructura de la casa domótica.....	117
Figura 28 Funcionamiento de la casa domótica.....	118
Figura 29 Estructura general de la arquitectura implementada.....	120
Figura 30 Arquitectura final implementada.	122
Figura 31 Casa domótica de Key-Studio.	123
Figura 32 Sensores de la casa domótica.....	124
Figura 33 Actuadores de la casa domótica.....	125
Figura 34 Estructura interna de la casa domótica.....	126
Figura 35 Arduino UNO y Sensor Shield.	128
Figura 36 Conexiones de los componentes electrónicos.	129
Figura 37 Esquemático del circuito.....	131
Figura 38 Funcionamiento del programa de Arduino.	133
Figura 39 Arduino IDE y conexión.	133
Figura 40 Controles de la aplicación de verificación.	137
Figura 41 Diagrama de flujo de las peticiones HTTP de la aplicación de verificación. .	139
Figura 42 Arc1hivo principal de la aplicación de verificación	141
Figura 43 Estructura final del backend.....	141
Figura 44 Estructura de un microservicio de un componente ciber físico.....	143
Figura 45 Relación entre microservicios.....	144
Figura 46 Estructura del paquete de la aplicación de un microservicio.....	147
Figura 47 Estructura del paquete del controlador de un microservicio.....	148
Figura 48 Estructura del paquete DAO de un microservicio.....	149
Figura 49 Estructura del paquete de entidad de un microservicio.....	150

	19
Figura 50 Estructura del paquete del servicio de un microservicio.....	152
Figura 51 Estructura del paquete de recursos de un microservicio.....	153
Figura 52 Estructura del microservicio Zuul.....	154
Figura 53 Clases del servicio Zuul.....	155
Figura 54 Estructura de la base de datos de un microservicio.....	156
Figura 55 Despliegue del agente Consul.....	157
Figura 56 Interfaz del agente de orquestación y descubrimiento.....	158
Figura 57 Estado de un microservicio en el agente de descubrimiento.....	158
Figura 58 Despliegue y funcionamiento del Broker.....	160
Figura 59 Instancias del microservicio.....	161
Figura 60 Publicación en el broker.....	162
Figura 61 Base de datos del microservicio.....	162
Figura 62 Instancia activa del microservicio.....	162
Figura 63 Registro de una instancia en la base de datos.....	163
Figura 64 Pulsador de encendido de las luces led.....	164
Figura 65 Base de datos.....	164
Figura 66 Estado actual del actuador.....	165
Figura 67 Estado actual del actuador.....	165
Figura 68 Estado apagado del actuador.....	166
Figura 69 Notificación de fallo de conexión.....	166
Figura 70 Prueba de funcionamiento de las peticiones y tiempo de respuesta.....	168
Figura 71 Peticiones de la prueba de funcionamiento.....	169
Figura 72 Prueba de carga con 100 usuarios.....	170
Figura 73 Pico de peticiones con 100 usuarios.....	170
Figura 74 Prueba de carga con 200 usuarios.....	171

	20
Figura 75 Pico de peticiones con 200 usuarios.....	171
Figura 76 Prueba de carga con 400 usuarios	172
Figura 77 Pico de peticiones para 400 usuarios	172
Figura 78 Prueba de carga con 800 usuarios	173
Figura 79 Pico de peticiones para 800 usuarios.	173
Figura 80 Prueba de carga con 1000 usuarios	174
Figura 81 Pico de peticiones para 1000 usuarios.	174
Figura 82 Prueba de carga con 5000 usuarios	175
Figura 83 Pico de peticiones para 5000 usuarios.	175

Resumen

En la actualidad han surgido enfoques importantes como el de la Web de las Cosas (WoT), convirtiendo a los servicios web en parte fundamental de la conectividad, y generando la necesidad de brindar sistemas escalables que permitan la gestión de múltiples componentes ciberfísicos. Los nuevos retos tecnológicos requieren la convergencia de los entornos físicos, virtuales y las herramientas capaces de manejar u orquestar todo al mismo tiempo. Bajo esta motivación, se propone un sistema domótico basado en la WoT, capaz de orquestar y descubrir componentes ciberfísicos y demostrar su aplicabilidad mediante una casa automatizada. Este sistema consta de cuatro capas principales, en la primera, se tiene el frontend, una aplicación de verificación programada en HTML, CSS y JavaScript, funcional en cualquier navegador web, la segunda es el backend, una arquitectura de microservicios desarrollada con SpringBoot, con bases de datos SQL, pero lo más importante, que funciona bajo un agente de orquestación y descubrimiento llamado Consul, con el cual se gestionan todos los microservicios de los componentes ciberfísicos, la tercera es un Broker de mensajería que funciona con el protocolo MQTT y la última capa es una casa domótica con múltiples sensores y actuadores, con un módulo ESP8266 como controlador, en donde se encuentra programada toda la lógica necesaria para publicar los datos importantes al Broker, para que los suscriptores en el backend puedan recibir estos datos y guardarlos, para que posteriormente puedan ser consultados por el frontend mediante API REST.

PALABRAS CLAVE:

- **WEB OF THINGS**
- **COMPONENTES CIBERFISICOS**
- **ORQUESTACIÓN Y DESCUBRIMIENTO**

Abstract

Nowadays, important approaches such as the Web of Things (WoT) have emerged, making web services a fundamental part of connectivity, and generating the need to provide scalable systems that allow the management of multiple cyber-physical components. The new technological challenges require the convergence of physical and virtual environments and tools capable of managing or orchestrating everything at the same time. Under this motivation, a WoT-based home automation system is proposed, capable of orchestrating and discovering cyber-physical components and demonstrating its applicability through an automated house. This system consists of four main layers, in the first one, we have the frontend, a verification application programmed in HTML, CSS and JavaScript, functional in any web browser, the second is the backend, a microservices architecture developed with SpringBoot, with SQL databases, but most importantly, that works under an orchestration and discovery agent called Consul, with which all the microservices of the cyber-physical components are managed, the third is a messaging Broker that works thanks to the MQTT protocol and the last layer is a domotic house with multiple sensors and actuators, with an ESP8266 module as controller, where all the necessary logic is programmed to publish the important data to the Broker, so that the subscribers in the backend can receive this data and save them, so that later they can be consulted by the frontend thanks to API REST.

KEYWORDS:

- **WEB OF THINGS**
- **CYBER-PHYSICAL COMPONENTS**
- **ORCHESTRATION AND DISCOVERY**

Capítulo I

Marco Metodológico

Antecedentes

Las TICs avanzan a pasos agigantados día tras día, y algo que se ha tornado muy popular e indispensable al hablar sobre tecnologías de la información, es el IoT. La primera concepción de IoT data en el año 1982 donde una máquina de coca cola fue modificada, brindando la posibilidad de reportar el número de bebidas que contiene y si estas están frías o no (Muhammad & Sadia, 2015). El IoT tiene como fin principal darnos el control de las cosas que nos rodean unificando todo bajo la misma estructura, además de informarnos del estado de estas (Sommaya, Ramaswamy, & Siddhart, 2015). El intercambio de información debe ser autónomo y cada dispositivo presenta una única identificación, este intercambio se da por tecnologías de radio frecuencia y redes inalámbricas de sensores (Muhammad & Sadia, 2015). IoT tiene infinidad de aplicaciones debido a que las necesidades cada día aumentan y resulta necesario tornar en inteligentes a diferentes campos como: agricultura, transporte, redes, seguridad, medicina y el ámbito residencial para las casas (Chen, Xu, Liu, Hu, & Wang, 2014).

La automatización en el ámbito residencial se convirtió en una ciencia, llamada domótica. La domótica nació como un nuevo ámbito ingenieril en el año 1978, al pasar de los años las empresas de electrodomésticos se vieron obligadas a incluir funciones inteligentes para suplir las necesidades del hogar, al pasar del tiempo se ha logrado integrar distintas tecnologías en casa con el uso de las telecomunicaciones, electricidad e informática (Herrera Quintero, 2015). La automatización residencial representa un gran escenario para el uso de internet of things (IoT), debido a que poseen muchos

componentes domésticos para la electrónica de consumo, generando un gran potencial de conexión mediante IoT (Salman, Salman, Jahangirian, & Abraham, 2016).

Internet de las Cosas (IoT) junto con el creciente número de dispositivos que cambian información a diario mediante internet dio paso a uno de los más grandes adelantos y tendencias en el campo investigativo, el Web of Things (WoT), que se define como una especialización de IoT que posibilita a cualquiera interconectar dispositivos por medio de la web y admite los estándares web abiertos para compartir información e interoperar dispositivos (Zeng, Guo, & Cheng, 2011). IoT significa darle conexión al mundo físico, dispositivos, a través del internet con el fin de optimizar tareas comunicando los componentes ciber-físicos (CPS) entre sí (Guinard & Trfia, 2016); mientras que WoT estrecha la barrera entre lo físico y lo virtual, ya que a cada “cosa” la convierte en una web thing o servicio web, teniendo de este modo un repositorio virtual de los elementos físicos existentes.

WoT, tal como el modelo OSI, aunque enfocándose a la capa de aplicación de este, maneja su estructura por capas, teniendo cuatro bien diferenciadas, la primera es la capa de acceso, encargada de convertir cualquier componente CPS (componente ciber-físico) en un servicio web, la segunda capa corresponde al descubrimiento que permite localizar a los servicios web y tornarlos usables para cualquier aplicación WoT creando al mismo tiempo una red de servicios web; la capa tres corresponde a la comunicación, la cual se responsabiliza de especificar que la transferencia de información entre los servicios web y el hardware sea segura y eficiente, finalmente, la capa de creación es la que interactúa con el usuario y se encarga de la integración de todas las capas para presentar una variedad de herramientas web para que el usuario pueda manejarlas como widgets (pequeñas aplicaciones) de fácil comprensión. Web of Things representa un gran avance y una herramienta importante para la consecución de

un sistema optimizado, seguro, flexible y escalable para el almacenamiento e intercambio de datos (Dominguez, 2007).

WoT se enfoca el desarrollo de servicios web debido a su funcionalidad entre aplicaciones sobre una red y la permisibilidad de intercambio de datos independientemente de las plataformas que se usen. Los servicios web son adjudicados como un caso especial de la arquitectura orientada a servicios (SOA) (Morales C. , 2010). Una arquitectura (SOA) se focaliza en el descubrimiento e integración de servicios, refiriéndose descubrimiento a la disponibilidad del catálogo de servicios con la seguridad necesaria para no sufrir ataques. Mientras tanto, la integración de servicios permite la transferencia de datos y la comunicación de los servicios web entre sí, generalmente se maneja por orquestación (un servicio web principal controla a los demás).

WoT se puede desenvolver en varios escenarios al igual que IoT, sin embargo, uno de gran utilidad y siendo más común es la domótica. Al contener gran variedad de componentes ciber-físicos en una casa Web of Things ayuda a crear un repositorio de todos estos haciéndolos parte de la web para que su uso sea más sencillo. Mediante Web of Things un hogar inteligente puede manejar un esquema con flexibilidad sin depender de software, topologías o plataformas propias de domótica ya que es posible crear un hardware de propósito general basado en plataformas de bajo costo como Raspberri-pi que permita interconectar los componentes CPS y controlarlos como servicios web que a posteriori serán presentados en una interfaz HMI para el manejo de los usuarios finales. De esta manera, se logran conectar actuadores y sensores al hardware creado, y con tecnología WoT se obtienen mayores beneficios ya que se maneja la integración de los mismos mediante orquestación e incluso al manejar el mismo lenguaje (característica propia de WoT) se pueden intercambiar componentes.

La Tabla 1. describe un resumen en el cual se puede ilustrar las partes más importantes de los trabajos de tesis desarrollado previamente en la Universidad de las Fuerzas Armadas ESPE y artículos científicos presentados en jornadas IEEE que guardan relación en ciertos aspectos con el proyecto propuesto. Se percibe que la tabla contiene cuatro columnas para especificar el artículo o tesis, con que tipos de tecnología se trabaja (IoT o WoT), que plataforma se utiliza y que tecnología adicional incorpora. Los cinco trabajos recolectados que se exhiben coinciden en el uso del internet de las cosas (IoT), mientras que tres de los mismos incursionan con tecnología de Web of Things (WoT), recalcando la importancia del uso de este tipo de tecnología en ámbitos con escenario doméstico.

Tabla 1

Trabajos relacionados

Proyecto	Uso de Internet of Things (IoT)	Uso de Web of Things (WoT)	Plataforma	Otros
Diseño e implementación de una interfaz inteligente para la gestión de recursos en el hogar mediante Smart TV, a partir de modelos y servicios. (Salazar, 2017) .	Si	Si	Arduino	Smart TV
Diseño e implementación de una Smart-meter de energía eléctrica enlazado en una plataforma de visualización para monitoreo y control del consumo energético domiciliario basado en IoT (Chávez & Herrera, 2020) .	Si	No	Raspberry-Pi	ESP8266 (Módulo WiFi para Arduino)

IoT based electrical device surveillance and control system. (Alok & Rahul, 2019).	Si	No	Raspberry-Pi	NodeMCU (Placa orientada a IoT)
A car as a Semantic Web Thing: Motivation and Demonstration (Klotz, 2018).	Si	Si	Raspberry-Pi	OBD (Sistema de Diagnóstico a Bordo)
A car as a Semantic Web Thing: Motivation and Demonstration (Klotz, 2018).	Si	No	Raspberry-Pi	Rapid Application Development (RAD) (Modelo de desarrollo de aplicaciones) Cloud based application)

Justificación e Importancia

Si bien la cuarta revolución industrial se maneja en base a la pirámide de automatización y de manera principal en sus dos últimos niveles, MES y ERP, lo que hace posible la integración de sistemas y tecnología es el “internet de las cosas”. El Internet de las cosas (IoT) es un esquema bien definido de estrategias informáticas interconectadas, dispositivos digitales y mecánicos que tienen la función de transmitir datos por medio de la red definida sin tener colaboración humana en ningún nivel (Salman, Salman, Jahangirian, & Abraham, 2016).

El término general de Internet de las Cosas (IoT), se ha focalizado de forma primordial en brindar conectividad en una variedad de espacios de red, restándole atención al funcionamiento de los dispositivos inteligentes. Aunque fue difícil inicialmente, IoT está bien posicionado en la actualidad tanto en la industria como en el día a día de las personas con el uso de estos smart devices, adicionalmente, se ha

generado un avance favorable en la construcción de modelos de interacción e interconexión escalables enfocados en la capa de aplicación, lo que se conoce como “Web of Things” (WoT). La arquitectura orientada a servicios (SOA) representa una gran ayuda al momento de gestionar datos en la nube, WoT junto con el uso de arquitecturas SOA, principios de adaptabilidad, escalabilidad e interfaces ha hecho posible un crecimiento tecnológico y la evolución de la web (Li & Chou, 2015).

Sin lugar a duda, las tecnologías de la información acompañadas con el desarrollo de IoT es pan de cada día en la actualidad y por esta razón varios conocedores han mencionado que los datos son el nuevo petróleo, debido a que estos son casi infinitos, acumulativos, reusables y se pueden utilizar al mismo tiempo (Chavez & Herrera, 2020). El internet de las cosas pretende ayudar a las personas en sus actividades cotidianas, siendo un escenario muy usual en la implementación de IoT la automatización residencial, es decir el ámbito domótico con componentes de uso doméstico para la electrónica de consumo. Sin embargo, no se ha podido generalizar el uso de IoT para proyectos domóticos debido al cambio de arquitectura para cada nuevo sistema, como obstáculo principal (Ferreira, 2014).

Web of Things (WoT) surge como un paso más allá del Internet de las cosas, presentando características importantes como la interconectividad que permite el acceso a la información y la comunicación, además de la escalabilidad al soportar la interconexión de varios dispositivos generando una red de información. Además, uno de los objetivos primordiales de WoT es resolver la interoperabilidad entre plataformas y con la ayuda de la arquitectura orientada a servicios (SOA) logran el propósito generando servicios web que se pueden interconectar y comunicar entre sí debido al uso de lenguajes comunes. La domótica se ha visto beneficiada con el desarrollo de WoT debido a que con su uso los componentes ciber-físicos de un hogar pasan a

convertirse en parte de la web, logrando tener acceso a estos por medio de un repositorio de forma dinámica, además de contar con una estandarización de mecanismos de comunicación entre hardware y software y no depender siempre de los mismos componentes, ya que con el uso de Web of Things el cambio de componentes es posible sin mayor cambio en el esquema general.

En tal sentido, este proyecto integra la tecnología WoT en el desarrollo e implementación de un esquema flexible enfocado en un escenario doméstico, para esto se propone la integración de sistemas ciber-físicos (CPS) representados por sensores y actuadores incorporados a una maqueta, los mismos que después de ser integrados a la web como servicios web generarán una red WoT que pueda ser controlada por un usuario final mediante una interfaz HMI. La consecución del sistema está dividida en dos fases, Back-End (acciones de lógica y desarrollo que no puede controlar el usuario final) conformada por los componentes CPS en su fase de conexión y de conversión a servicios web con la ayuda de un Arduino como base para la creación de un nuevo hardware de propósito general. A la vez, la segunda fase se refiere al Front-End (parte visual para el usuario final) que consta de un HMI (interfaz hombre máquina) que representa la orquestación de acciones dentro del sistema doméstico (Wang, Suparna, Zhou, Huang, & Moessner, 2017).

El sistema a implementarse, se presenta como un esquema flexible que usa a la domótica únicamente como escenario dado a que no requiere el uso de topologías o software propios de domótica ya que, como se ha mencionado previamente se diseñará e implementará un hardware en base a la plataforma Arduino (debido a sus prestaciones, flexibilidad y bajo costo) para que a partir de esta se permita integrar los componentes ciber-físicos y con el uso de conexión WiFi y web of things crear una red de servicios web. La integración de servicios web se ejecuta mediante orquestación, lo

que significa que existe un servicio web principal que controla las acciones de los demás y simultáneamente se produce la comunicación entre capas, hardware y servicios web se basa en lenguajes de uso común, en el caso actual JSON (formato de texto sencillo para el intercambio de datos, de lenguaje JAVA), por tal motivo, el hardware genera un JSON y al mismo tiempo los servicios web_esperan el mismo JSON o viceversa, dándose la comunicación de esta manera. Al hablar el mismo lenguaje, el cambio de componentes es permitido (Herrera Quintero, 2015).

Alcance del Proyecto

La integración entre hardware y software representa un gran paso en el desarrollo tecnológico del mundo actual, al igual que la automatización dentro de los hogares. En tal sentido, este proyecto pretende desarrollar e implementar un sistema basado en la Web of Things (WoT) teniendo como escenario la domótica, creando un prototipo mediante una maqueta con componentes ciber-físicos (sensores y actuadores), sin utilizar software de fabricante o protocolos domóticos debido a que se procederá a la creación de un hardware propio de propósito general teniendo como base el controlador genérico Raspberry-pi, logrando de este modo la integración de los componentes ciberfísicos (CPS) con la web para la electrónica de consumo utilizada en el hogar, es decir en el dominio de la domótica. Para esto, se presenta al sistema en dos capas, front-end (presentación visual al usuario final) con un HMI que permita el control de los CPS convertidos previamente en microservicios, que a su vez son interconectados vía web de primera mano en el back-end (capas de entrada a las que el usuario final no tiene acceso) con el uso del hardware de propósito general con conexión WiFi. Cabe recalcar que el proyecto tendrá un esquema flexible, lo que quiere decir que se integrarán los componentes (hardware con servicios web) a través de la

web of things debido a la facilidad de la comunicación entre capas y además se pueden cambiar los componentes ya que todos hablan el mismo lenguaje.

El proyecto se realiza por etapas, basadas en las cuatro capas del WoT (Web of Things) e integrando servicios web por orquestación (se tiene un servicio web principal que controla las acciones de los demás). Las dos etapas iniciales corresponden al back-end, en la primera etapa se tiene la incorporación de componentes ciber-físicos (actuadores y sensores) al controlador, con el uso de conexión WiFi se ingresa a la capa de acceso transformando estos componentes en web things (servicios web). La segunda etapa consiste en la capa de descubrimiento donde como su nombre lo indica, el propósito es buscar los servicios web existentes, para esta etapa las herramientas Consul y Springboot constituyen una gran ayuda trabajando a la par, debido a que después de ser generados los servicios web Consul además de ayudar en la generación y registro de los mismos también permite encontrarlos mientras que Springboot se encarga de su coordinación y que trabajen en el mismo ecosistema permitiendo que se interconecten.

Finalmente, se incursiona en el front-end teniendo como siguiente etapa la comunicación, en donde se tiene un repositorio dinámico de servicios web en el cual ya se puede realizar un intercambio de datos e información a través de internet, de manera segura y precisa, llegando a la última etapa que corresponde a un HMI que permite al usuario final interactuar con el repositorio de servicios (previamente componentes ciber-físicos) de manera sencilla y comprensible permitiendo el control de estos según la preferencia de cada usuario. El esquema general del proyecto se presenta en la Figura 1, donde se presenta todas sus capas y las tecnologías más importantes utilizadas.

Debido a que se diseñará e implementará un prototipo en maqueta, a continuación, se enlistan los elementos a ser usados en esta, reiterando que no resulta

necesario usar software, topologías o plataformas propias de la domótica debido a la creación de un propio hardware de propósito general usando a la domótica como escenario exclusivamente.

Sensores:

- Sensor de movimiento PIR.
- Sensor de gas MQ-2.
- Sensor de humedad del suelo.
- Sensor de vapor.
- Sensor zumbador pasivo.
- Sensor de fotocelda

Actuadores:

- Módulo Relé para encender y apagar las luces.
- Servo Motores.
- Indicador Led blanco.
- Indicador Led amarillo.
- Pantalla LCD-1602.

Plataforma de hardware:

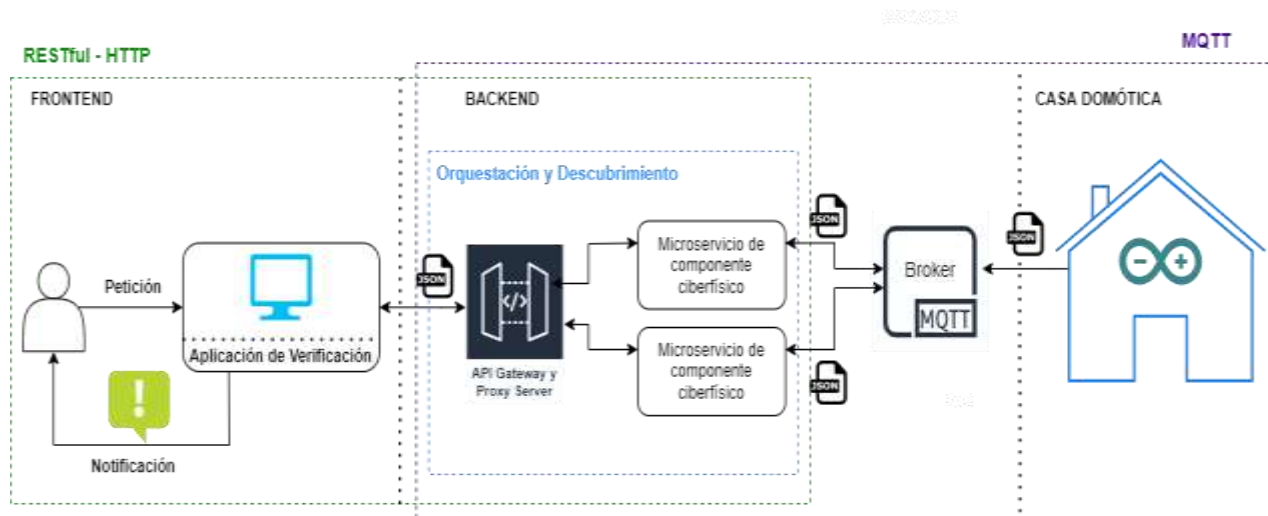
- Raspberry-Pi.

Tecnologías de Conexión:

- WiFi.

Figura 1

Esquema general del proyecto.



Objetivos

General

Diseñar e implementar un sistema domótico basado en la WoT para la orquestación y descubrimiento de componentes cyber físicos empleando esquemas de orquestación y descubrimiento.

Específicos

- Diseñar una arquitectura orientada a servicios aplicado al dominio de la domótica.
- Implementar un set de componentes ciber-físicos para el dominio de la domótica.

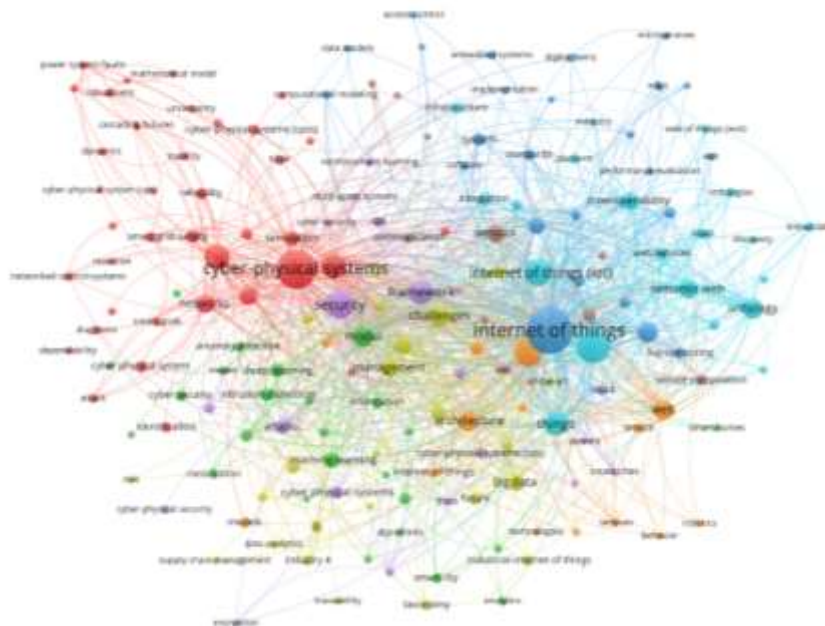
- Implementar los microservicios de los componentes ciber-físicos y los servicios de orquestación y descubrimiento para la lógica de negocio del sistema.
- Implementar un HMI basado en web para interactuar con los servicios de la lógica de negocio del sistema.
- Implementar una maqueta con los sensores y actuadores previstos para realizar pruebas de validación del sistema.
- Analizar el funcionamiento del sistema a través de pruebas en la maqueta.
- Ejecutar pruebas de usabilidad con diferentes usuarios para verificar la facilidad de uso del sistema, con escala SUS (Escala de Usabilidad de Sistemas).
- Analizar el comportamiento de los servicios web del sistema por pruebas de carga por medio del framework gatling.

Estado del Arte

Como introducción a esta sección, es importante conocer la densidad de las búsquedas que tienen las temáticas propuestas, por lo que se ha utilizado el software VOSviewer con la finalidad de crear mapas de información basados en los datos de la red. Se introdujo las palabras clave expuestas en el resumen: “web of Things”, “componentes ciberfísicos” y “orquestación y descubrimiento. En primer lugar, se obtuvo un mapa de la densidad de búsquedas por palabra, como se puede ver en la figura 2, y el resultado que más relevancia toma es el de “internet of things”, seguido por “sistemas ciberfísicos”. En segundo lugar, se obtuvo un mapa de la densidad de búsqueda por países, como se puede ver en la figura 3, y el resultado que más relevancia toma es, “Estados Unidos”, seguido por “China” y “Korea del Sur”.

Figura 2

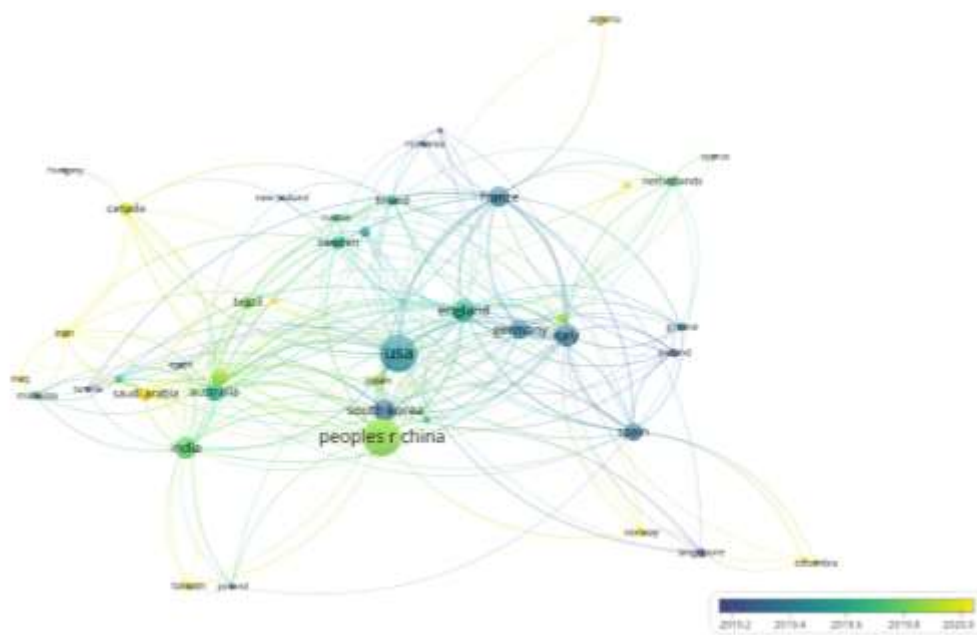
Densidad de búsquedas por palabras.



Nota: Figura generada por el software VOSviewer.

Figura 3

Densidad de búsquedas por países.

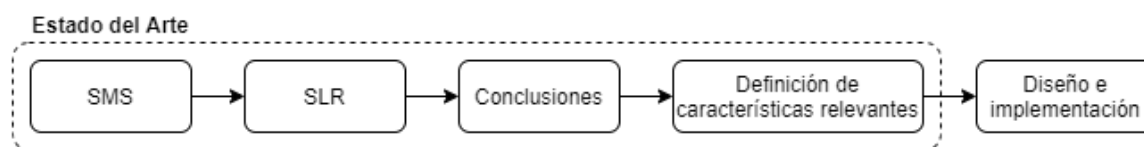


Nota: Figura generada por el software VOSviewer.

El estado del arte como tal, está dividido en dos secciones importantes. La primera sección corresponde a un *Mapeo Sistemático de la Literatura* (SMS, Systematic Mapping Study), utilizado para recopilar la información necesaria de una manera general sin ahondar tanto en los detalles. La segunda sección corresponde a una Revisión Sistemática de la Literatura (SLR, Systematic Literature Review), utilizada para recopilar la información de una manera mucho más específica y abordando temas de mayor importancia para el desarrollo de este trabajo. Una vez concluidas estas dos secciones se determina las conclusiones más relevantes y se define las características más importantes de toda la arquitectura para finalmente continuar con el diseño y la implementación. La figura 4 pone a la vista el proceso a seguirse para desarrollar el trabajo, dando inicio con el estado del arte.

Figura 4

Proceso de investigación del Estado del Arte.



Nota: Figura inspirada en (Moguel Márquez, 2018)

Síntesis de evidencia disponible

Mapeo Sistemático de la Literatura

Previamente se ha justificado la importancia de la Web of Things en los entornos actuales, orientándose a cumplir un papel crucial con todo lo relacionado en la industria 4.0 junto con paradigmas como SOA y específicamente los microservicios, además de

los sistemas ciber físicos como parte importante de la integración de un sistema domótico específico.

La interoperabilidad de tecnologías y herramientas como tal son cada vez más importantes, ya que, eso es lo que busca con paradigmas como el WoT, de aquí a unos años sin lugar a dudas muchos campos del comercio, la industria y la ciencia, tendrán bases importantes en paradigmas de este tipo.

Una de las grandes ventajas del mundo actual son las redes inalámbricas que permiten que los procesos de automatización se implementen de manera mucho más rápida y eficiente, además de garantizar altas velocidades de respuesta debido a los protocolos de comunicación actuales.

Por estas razones y para un conocimiento claro acerca de que herramientas y tecnologías utilizar en el desarrollo del proyecto para que cumpla con los requerimientos de ser un sistema domótico específico, con múltiples sensores y actuadores y sistemas ciber físicos que deben ser orquestados y descubiertos , se planteó un SMS (Kitchenham & Charters, 2007), con el cual de una manera muy general se especifican las posibles herramientas y tecnologías que cumplan con los requerimientos.

Como conclusión se puede mencionar que el SMS proporciona un macro conocimiento acerca de las temáticas puntuales para la realización de este trabajo de una manera adecuado, inclusive para futuros trabajos o investigaciones.

Objetivos. En los estudios existentes no se aborda ampliamente el paradigma de la Web of Things (WoT), pero si se trata acerca de temáticas con un poco más de relevancia como es la domótica, pero estos están orientados a problemáticas concretas, por lo que no se tiene todas las temáticas en una sola investigación, si no en varias.

No hay que negar qué por la innovación y los estudios realizados, se ha logrado un gran avance en campos de la automatización como son la domótica y los sistemas ciberfísicos, parecidos a IoT, pero aún falta mucha investigación sobre todo en temáticas puntuales como son la orquestación y descubrimiento de componentes a través de servicios y tecnologías en entornos virtuales, para garantizar rapidez y eficiencia en aplicaciones orientadas a la automatización de todo tipo y en este caso de hogares inteligentes.

La relevancia del SMS radica en las bases claras que proporciona para lograr conocer los campos de análisis e indagación recientes, y saber si esa documentación, trabajos o artículos académicos tienen la posibilidad de ser replicables e implementados en espacios como el planteado de un hogar inteligente con base en WoT para orquestar el funcionamiento de sistemas ciber físicos. También brindo un panorama más extenso acerca de ideas innovadoras y nichos estratégicos de trabajo que pueden ser explotados en el futuro, pero sobre todo que brinden la confiabilidad tanto por la parte del cliente como por parte del proveedor, buscando siempre robustez, escalabilidad e interoperabilidad entre herramientas, tecnologías y también porque no, entre usuarios.

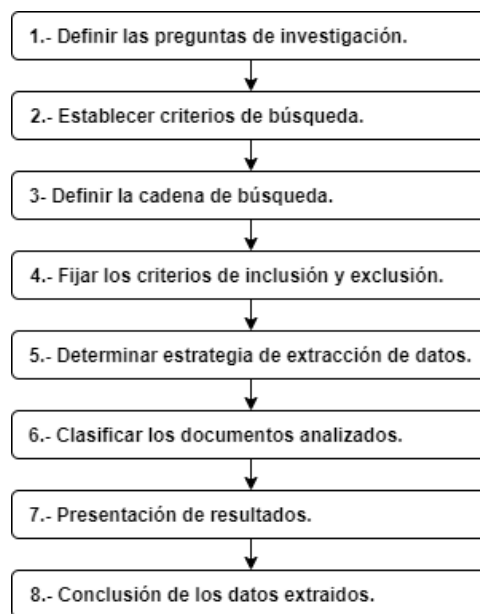
Como punto especial del SMS es la facilidad y versatilidad al momento de plantear las preguntas de investigación que pueden utilizarse para el proceso de investigación y que garantizan abarcar todos los campos de estudio y publicaciones vanguardistas de las temáticas tratadas y en este caso, la domótica, los sistemas ciberfísicos, las tecnologías virtuales que cumplen con el papel de orquestación y descubrimiento de sistemas ciberfísicos, y finalmente la integración de todos estos componentes en el paradigma WoT. La información, datos y resultados otorgados por este estudio fueron muy importantes como punto de inicio para las futuras investigaciones.

Método de investigación. Un SMS se basa en una metodología donde se prioriza la categorización, clasificación y estudio de todos los temas de forma extensa a fin lograr el razonamiento que ya existe en estas superficies de análisis (Kitchenham & Charters, 2007; Petersen, Feldt, Mujtaba, & Mattsson, 2008). Luego de contestar las cuestiones planteadas para el análisis, se hace a la clasificación de la información obtenida, de este modo se ahorra esfuerzo y mucho tiempo en la investigación. Para lograr todos y cada uno de los objetivos, el SMS debe ser elaborado con rigor, de la manera más completa y con una calidad lo más óptima posible. En cada uno de los puntos que conforman el SMS, radica la importancia de usar este instrumento, ya que de esta manera se puede identificar cada uno de los temas que se desea y siempre obtenido una cantidad de información suficiente, que posteriormente llevara a revisiones sistemáticas y estudios innovadores (Kitchenham, Budgen, & Brereton, 2011).

Al SMS se lo adapto de tal manera que vaya de acuerdo a los procesos de estudio del trabajo, además se utilizó las herramientas y técnicas pertinentes, ya que, lo que se pretende es abordar tanto los ámbitos de la automatización, el control y la domótica en el entorno físico, así como el uso de tecnologías de orquestación, descubrimiento y manejo de dispositivos físicos a través de un entorno virtual. Todo el procedimiento realizado es exhaustivo y específico, además cada uno de los pasos a seguir está bien definido, en la figura 5, donde se presenta el proceso de elaboración del SMS, aplicando también ciertas consideraciones y recomendaciones (Bailey, y otros, 2007).

Figura 5

Proceso de elaboración del SMS



Nota: Inspirada en (Moguel Márquez, 2018). Tomada de (Cobo Jaramillo, 2021).

Definir preguntas para la investigación. Ya que como objetivo se tiene el examinar las tecnologías y herramientas más actuales para sistemas domóticos como el propuesto, incluyendo tanto el entorno físico que se automatizará y controlará, así como también el entorno virtual encargado de orquestar y gestionar cada una de las acciones que se quiere realizar, se planteó las preguntas específicas presentadas en la tabla 2.

Tabla 2

Preguntas de investigación para desarrollar SMS

Pregunta de investigación	Motivación
Q1: ¿Qué tecnologías y/o técnicas se pueden para la implementación de sistemas domóticos?	Conocer qué tecnologías, librerías específicas de desarrollo y pluggins adicionales existen, y que tan desarrolladas están.

Q2: ¿Qué tecnologías y/o técnicas se pueden usar para el desarrollo de servidores o servicios específicos para la orquestación y descubrimiento de componentes?	Conocer las tendencias de desarrollo más utilizadas tanto para componentes de hardware como de software.
Q3: ¿Qué herramientas y/o dispositivos son los más óptimos en un sistema orientado a la WoT donde se cuente con sistemas ciberfísicos?	Saber cuáles son los hardware específicos existentes que tienen la posibilidad de utilizarse en sistemas domóticos orientados a la WoT y que logre ser controlados y orquestados adecuadamente.
Q4: ¿Qué tecnologías y/o técnicas existen para obtener, procesar y almacenar los datos involucrados con el control y automatización de sistemas ciber físicos en un sistema domótico?	Conocer las tecnologías que existen para el manejo y la gestión correcta de los datos utilizados en un entorno domótico orientado a la WoT.

Establecer los criterios de averiguación primarios y secundarios. Para la recopilación de los artículos científicos pertinentes con las temáticas propuestas, las librerías digitales utilizadas fueron IEEE Xplore, MDPI, Springer Link y Science Direct. Esta decisión fue tomada en base a la gran disponibilidad y amplia cobertura en todos los campos de la ciencia que todos estos catálogos ofrecen, pudiendo abordar sin problema ámbitos como la domótica.

Al tratarse de un mapeo general, no se excluye ningún tipo de artículo, ya que se quiere tener una idea global acerca de todas las técnicas, tecnologías y herramientas más utilizadas y su desarrollo con respecto a las necesidades actuales. Es por ello que se tendrá en cuenta todos los documentos independientemente del origen de los mismos.

Definir las cadenas de búsqueda. Al no conocer el volumen total de la información que podría recopilarse, en un principio se realiza una exploración general para determinar la cantidad de recursos aproximada, por ello se determinó la

importancia de definir ciertas cadenas de búsqueda y su relevancia para la tesis. Para poder definir las cadenas de averiguación en el SMS, se usa las cuestiones de indagación presentadas en la tabla 2 y está establecido los términos más relevantes presentados en la tabla 3, que van a ser usados en idioma inglés dado que es el estándar de las librerías digitales.

Tabla 3

Términos utilizados en las cadenas de averiguación del SMS.

Término principal	Términos alternativos
Domotics. [Q1]	"Home automation" OR "Smart homes".
Orchestration and Discovery [Q2]	"Service Orchestration" AND "Service Discovery"
Cyber-physical system. [Q3]	CPS.
WoT [Q3, Q4]	"Web of Things" OR IoT.
Automation and Control [Q4]	Automation AND Control

Debido a que con una cadena general uniendo todos los términos, no se obtendría resultados prometedores para la realización de este SMS. Se decidió realizar un total de tres búsquedas en paralelo con el fin de ampliar el panorama de estudio y obtener mayor cantidad de recursos y de información y para esto se utilizó las siguientes cadenas de búsqueda:

1. Cadena de búsqueda 1:

("Domotics" OR "Home automation" OR "Smart homes".)

AND

("Orchestation and Discovery" OR "Service Orchestation" OR "Service Discovery".)

AND
 ("Cyber-physical system" OR CPS).

2. Cadena de búsqueda 2:

("Domotics" OR "Home automation" OR "Smart homes".)
 AND
 ("Orchestation and Discovery" OR "Service Orchestation" OR "Service
 Discovery".)
 AND
 ("Web of things" OR WoT OR IoT).

3. Cadena de búsqueda 3:

("Domotics" OR "Home automation" OR "Smart homes".)
 AND
 ("Orchestation and Discovery" OR "Service Orchestation" OR "Service
 Discovery".)
 AND
 ("Automation and Control" OR Automation AND Control).

Establecer juicios de inclusión y exclusión. Un dato importante que
 mencionar es

qué se filtró las fechas de publicación únicamente hasta agosto de 2021, y desde enero
 de 2016, asimismo debe tomarse en cuenta que después de la recopilación de la
 información y resultados de este SMS, se procede con una revisión sistemática de la
 literatura para abordar temáticas más específicas.

Al tratarse de una porción considerable de documentos encontrados, se
 presentó la necesidad de delimitar ciertas condiciones extra, aparte de la fecha que los
 documentos tienen que llevar a cabo para ser integrados:

- No puedes ser artículos incompletos

- Pertenecientes a los campos ya sea de “Ciencias de la computación / Computer Science” o “Ingeniería / Engineering”, con relación al ámbito tecnológico.
- Cualquier clase de documentación, publicación o investigación, sean estos tesis, conferencias, revistas, artículos, etc.
- Estudios que presenten de una manera clara el proceso de implementación y la estructura específica de implementación domótica.
- Estudios que no sean solo estudios del arte, sino que ya tengan cierto desarrollo en cualquiera de las temáticas propuestas.
- Estudios que incluyan las temáticas propuestas y sobre todo estén orientadas al campo de la domótica, pero especialmente en hogares inteligentes, ya que el enfoque del trabajo es ese y no se quiere desviar la terminología o la funcionalidad a campos diferentes como la industria.

Determinar la estrategia de extracción de datos. Una vez se tiene toda la información y todos los documentos recopilados, es necesario sintetizar los datos, con ciertas consideraciones que se menciona en la guía del SMS, debido a su extensión y ya que, no es de gran importancia para el desarrollo de este trabajo detallar dicho proceso, solo se lo realizó y se pudo registrar toda la información relevante, ya sea el campo de aplicación, el enfoque que se le da a la investigación y finalmente se eliminó los artículos repetidos y de poco interés para el trabajo.

Clasificación de los artículos. Para el proceso de obtención de los artículos de los cuales posteriormente se extraerá la información, se debe seguir tres fases importantes, en las cuales se va definiendo diferentes filtros de búsqueda:

1. La primera fase consiste en utilizar las cadenas de búsqueda expuestas anteriormente y en los motores de búsqueda únicamente considerar: **Title + Keywords + Abstract**, de esta manera se obtuvo una cantidad total de 13977 resultados entre las diversas bibliotecas digitales, de la forma siguiente:
 - **IEEE Xplore:** 2377 resultados.
 - **Springer Link:** 1257 resultados.
 - **ScienceDirect:** 559 resultados.
 - **MDPI:** 9784 resultados.

2. La segunda fase consiste en utilizar las mismas cadenas de búsqueda, pero además aplicar los criterios de inclusión y exclusión, y finalmente en los motores de búsqueda de cada librería utilizar: **Title + Keywords + Abstract**, obteniendo de este modo un total de 354 documentos entre las bibliotecas digitales, de la siguiente manera:
 - **IEEE Xplore:** 145 resultados.
 - **Springer Link:** 89 resultados.
 - **ScienceDirect:** 67 resultados
 - **MDPI:** 53 resultados.

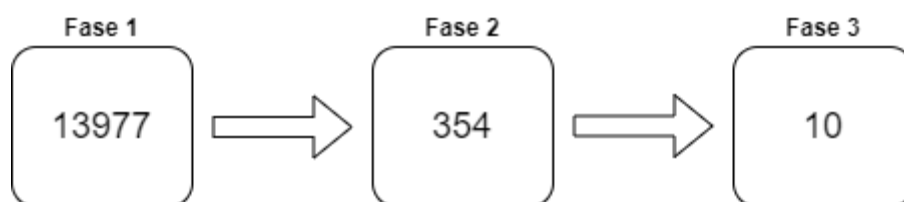
3. Para finalizar, la tercera fase, consiste en aplicar los mismos criterios utilizados en las fases anteriores y adicionalmente utilizar los motores de búsqueda de una manera más exhaustiva usando los filtros: **All Metadata**, en otras palabras, **Title + Keywords + Abstract + Full Article**, y además prestando mucha atención en el contenido importante de los artículos en sí, de este modo se ha obtenido un total de 10 documentos importantes y con relevancia para el trabajo, su distribución en las diferentes bibliotecas digitales es la siguiente:

- **IEEE Xplore:** 5 resultados.
- **Springer Link:** 2 resultados.
- **ScienceDirect:** 2 resultados
- **MDPI:** 1 resultados.

Asimismo en la figura 6, la cantidad total de artículos extraídos en cada una de las fases es presentada:

Figura 6

Cantidad total de artículos en cada fase del SMS.



Resultados. Como se observa, una vez elaborado todo el proceso planteado en el SMS, se obtuvo 10 artículos completamente relevantes y positivos para este trabajo. Dentro de los datos y la información que se pudo recopilar elaborando este SMS, lo más importante es sin duda la gran cantidad de información disponible respecto a las temáticas propuestas, lo que refleja el gran interés en estos campos de investigación, además de la gran aplicabilidad que seguramente ya se están presenciando, sobre todo en la rama de la domótica, ya que, cada vez más se busca la comodidad y que mejor que disponer de una casa inteligente, donde se pueda utilizar cada uno de los dispositivos disponibles a través de la red y desde la palma de la mano.

Conclusiones del SMS. Para poder obtener las conclusiones lo más claras e ilustrativas posibles, es importante responder todas las preguntas que fueron realizadas anteriormente y que sirvieron de motivación para poder realizar este mapeo de una manera mucho más profunda y que ahora pueden ser contestadas utilizando toda la información obtenida en este estudio, adicionalmente se puede mencionar que también se obtuvieron temáticas extras que servirán para la elaboración de la revisión sistemática posterior, que es de gran importancia para consolidar todas las temáticas necesarias para un producto final adecuado.

Q1: ¿Qué tecnologías y/o técnicas existen para la implementación de sistemas domóticos?

Después de todos los estudios realizados y con la información recopilada acerca de sistemas domóticos, las tendencias más utilizadas para las casas inteligentes actuales van enfocadas a ser accesibles para cualquier persona, inclusive con algún tipo de discapacidad, de este modo es que se presentan propuestas donde se utiliza tecnologías como Google Assistant, el asistente virtual propio de dicha empresa que está desarrollado con inteligencia artificial y que se encuentra en muchos dispositivos móviles, como celulares, parlantes inalámbricos, etc. (Roy, 2021; Rahman, 2019), su uso puede ser mediante texto o voz. También se menciona asistentes propios desarrollados con alguna librería de inteligencia artificial open Source, pero se les da enfoques relacionados directamente con la voz, esto con el fin de evitar la manipulación de dispositivos, ya que por lo general las personas mayores no comprenden del todo su funcionamiento (Jain, 2019; Rahman, 2019).

Dentro de las plataformas que se presentan, se encuentra Blynk, relacionada directamente con IoT y dispositivos en la nube, su funcionamiento consiste en la

conectar los dispositivos a la nube y el servicio se encarga de su funcionamiento, sin embargo, el inconveniente que presenta es que solo se puede utilizar si se contrata algún plan de suscripción que ofrece la compañía. (Alsuhaym, Al-Hadhrami, Saeed, & Awuson-David, 2021; Roy, 2021).

Finalmente se menciona los algoritmos que más versatilidad le podrían dar a un hogar inteligente específico, entre los cuales por ejemplo se encuentra el control Fuzzy, que consiste en definir los términos, variables y reglas que relacionan la entrada con la salida del sistema. (Nur Aziza, Siswipraptini, Jabbar, & Ruli Siregar, 2021) También hay enfoques especializados en HVAC, esto con el fin de garantizar una climatización adecuada dentro de los hogares con respecto al ambiente exterior (Roy, 2021). Y por último los algoritmos de machine learning y los algoritmos ML, orientados al aprendizaje propio de la máquina y a la gestión de los datos de maneras específicas, esto con el fin de garantizar un producto final adaptado a las necesidades de cada usuario. Dentro de estos algoritmos se encuentran: K-nearest neighbors, Decision Tree Regression, Support Vector Regression y Random Forest (Raju, 2021).

Q2: ¿Qué tecnologías y/o técnicas existen para el desarrollo de servidores o servicios específicos para la orquestación y descubrimiento de componentes?

En la mayoría de artículos se menciona la utilización de servicios web para la gestión de los componentes, tanto para orquestar, registrar y descubrir nuevos dispositivos, pueden tratarse ya sea de servicios locales, como también de servicios en la nube, cuando se trata de servicios en la nube se menciona a AWS, Firebase, especializados en albergar productos de este tipo, y cuando se trata de servidores locales se mencionan algunos desarrollados bajo lenguajes de programación como

PHP, adaptado especialmente al desarrollo web (Mahamud, 2019; Haque, 2019), JavaScript, con dinamismo innato y especializado en aplicaciones FrontEnd, también se mencionan lenguajes de programación orientados a objetos como C embebido (Agarwal, 2019), C# (Jain, 2019) y también Java, y este último con un Framework específico llamado Spring que dispone de una gran cantidad de herramientas y librerías especializadas en microservicios, que son registrados y orquestados. (Haque, 2019). Además, también se mencionan otro tipo de softwares como ThingSpeak, el cual facilita acceder, recuperar y registrar los datos utilizando su propia API (Alsuheyam, Al-Hadhrami, Saeed, & Awuson-David, 2021).

Q3: ¿Qué herramientas y/o dispositivos son los más óptimos en un sistema orientado a la WoT donde se cuente con sistemas ciberfísicos?

Los sistemas ciberfísicos son parecidos a un dispositivo IoT, con la diferencia de que no necesariamente necesitan estar conectados al internet para poder funcionar, dentro de estos se puede incluir una gran cantidad de sensores, como por ejemplo de luz, de humo, de temperatura, de movimiento, etc. (Mustafa, 2021; Mahamud, 2019; Jain, 2019), y también actuadores como por ejemplo Relés (Haque, 2019) y Motores, ya sean Servos o DC (Rahman, 2019). Además, se debe contar con Optoacopladores y TRIAC para garantizar la confiabilidad, ya que, se está trabajando con los voltajes que se tiene normalmente en los hogares de 100V a 240V (Agarwal, 2019).

Una parte fundamental para poder controlar de manera adecuada todos estos dispositivos, es el microcontrolador, que puede ir desde memorias EEPROM hasta plataformas más sofisticadas como son la Raspberry Pi, una computadora portátil (Nur Aziza, Siswipraptini, Jabbar, & Ruli Siregar, 2021; Raju, 2021), el Arduino en cualquiera de sus modelos, UNO, MEGA, NANO, etc. (Mustafa, 2021; Mahamud, 2019; Jain, 2019;

Agarwal, 2019), y el ESP32 (Roy, 2021; Mahamud, 2019) o el ESP8266 NodeMcu, basados en Arduino pero con interfaces específicas propias, y orientadas a IoT usando su módulo WiFi, que puede conectarse a Internet desde cualquier tipo de red (Haque, 2019).

Q4: ¿Qué técnicas y/o tecnologías existen para adquirir, procesar y almacenar los datos relacionados con la automatización y control de sistemas ciber físicos en un sistema domótico?

Los datos en cualquier sistema deben provenir de alguna variable a medir y se debe contar con algún dispositivo que mida dicha variable, de este modo se debe tener muy claro que para la adquisición de datos se necesita sensores específicos para cada tipo de variable y también se los puede introducir manualmente, de este modo se tiene elementos como Sonares, Radares, etc. (Rahman, 2019). Para la transmisión de los datos puede ser directamente por cables, comunicación serial o también comunicación inalámbrica como Bluetooth o IR (Rahman, 2019). Con esto en mente ya se puede proseguir con el procesamiento de los datos, según los estudios realizados se los realiza en los microcontroladores como son la Raspberry-Pi (Nur Aziza, Siswipraptini, Jabbar, & Ruli Siregar, 2021) o si se los envía a un computador directamente pueden utilizarse Softwares como TensorFlow, especializado en aprendizaje automático y que pueden manejar algoritmos como KMeans (Raju, 2021; Nur Aziza, Siswipraptini, Jabbar, & Ruli Siregar, 2021). Finalmente, para el almacenamiento de los datos, por lo general se utiliza bases de datos locales, que pueden ser relacionales como MySQL o bases de datos no relacionales como MongoDB (Mahamud, 2019), también se puede utilizar bases de datos en la nube como Firebase o las que

brindan los servicios como AWS o Heroku (Nur Aziza, Siswipraptini, Jabbar, & Ruli Siregar, 2021).

Otras conclusiones:

Después de haber solventado las preguntas de investigación, es pertinente agregar conclusiones extra que pueden ser de ayuda para el desarrollo del trabajo:

- Para la orquestación de componentes o de acciones que deban realizar los mismos, lo más conveniente es usar arquitecturas orientadas a servicios, SOA y más específicamente las llamadas arquitecturas en microservicios, ya que cuentan con librerías especializadas para estas tareas.
- Todo sistema o arquitectura debe tener una estructura adecuada y específica, de este modo es importante seguir los patrones o estándares ya definidos por instituciones como la IEEE.
- En el ámbito domótico es importante cubrir la mayor cantidad de requerimientos del usuario, pero es importante no descuidar una necesidad y poner mayor énfasis en otra, todas deben tener el mismo nivel de importancia.
- Uno de los puntos relevantes a nombrar es que los protocolos de comunicación más implementados en esta clase de aplicaciones son los ejecutados sobre TCP/IP, siendo de esta forma HTTP o MQTT, adicionalmente brindan una enorme versatilidad debido a que, el manejo de sus datos es ligero con HTML, XHTML, XML o JSON, siendo este último el mayormente utilizado por su fácil interacción con la mayoría de servicios web.

Como conclusión final se puede ir planteando la estructura que se busca con este trabajo, un sistema domótico versátil que cuente con múltiples sensores y actuadores los cuales van a ser orquestados por algunos servicios web y además como núcleo

tendrán un microcontrolador, en el cual se procesaran los datos para después ser almacenados en alguna base de datos. Es importante mencionar que acorde se avanza con el proyecto, algunos componentes pueden variar, pero la funcionalidad que se busca es la misma.

Revisión Sistemática de la Literatura

Una vez finalizado el SMS y con toda la información clasificada, surge la necesidad de profundizar aún más en ciertos campos de estudio y líneas de trabajo, ya que no se menciona mucho acerca de la importancia de las arquitecturas orientadas a servicios y más específicamente de los *microservicios*, y tampoco acerca del control en sí, y en este caso concreto acerca de la importancia de la tecnología WiFi u otras alternativas para el *control telemático* orientado a sistemas ciber físicos, es por ello que se procede con esta SLR.

La diferencia primordial entre SLR y el SMS, es que en el SLR se profundiza temáticas específicas y se aborda incluso propiedades más técnicas que en el SMS no eran tan importantes. Además, el propósito más importante que tiene es el de complementar la información del SMS y de este modo lograr culminar el estado del arte de manera adecuada.

Es por esto que en esta sección se desarrolla el SLR, para cumplir con el objetivo principal de brindar un conocimiento mucho más amplio en temáticas como son los *microservicios* y *control telemático*, ya que son pilar fundamental para este trabajo.

Objetivos. El objetivo general es profundizar en las técnicas, tecnologías, métodos y herramientas que se utilizan para poder orquestar, registrar y descubrir microservicios orientados a un sistema domótico y de igual manera poder conocer

cuáles son los medios o métodos de control telemático más óptimos en un entorno domótico local, donde es importante la velocidad de respuesta.

Toda la información que se logrará recopilar con este SLR, ayudará a solventar dudas necesarias para que el producto final pueda ser lo más amigable con el usuario posible y que le brinde una calidad de servicio óptima.

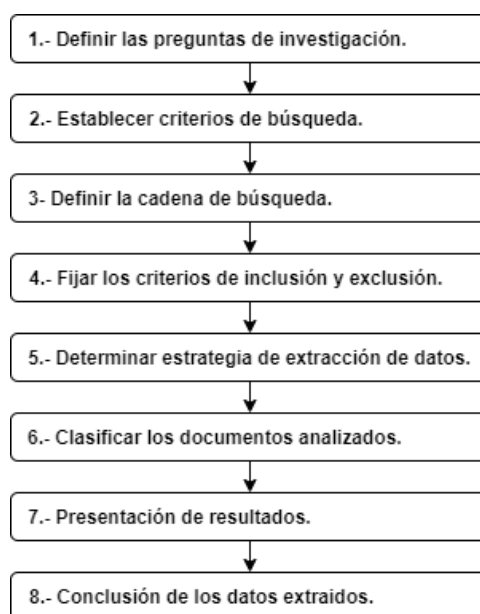
Finalmente, otra de las razones más importantes para escoger realizar un SLR, es que al igual que el SMS, dispone de preguntas que pueden ser utilizadas para filtrar las temáticas específicas y complementar la información que ya se encuentra disponible.

Método de investigación. El proceso para desarrollar un SLR, es similar al de un SMS, pero se enfoca más en tres actividades principales: plantear, ejecutar y extraer resultados. La fase de planteamiento consiste en elaborar el plan de trabajo junto con las temáticas importantes. La fase de ejecutar consiste en buscar y extraer toda la información pertinente y posteriormente pasará por ciertos parámetros y condiciones que permitirán seleccionarla y después sintetizarla efectivamente. Por último, la tercera fase consiste en la interpretación de toda la información para que pueda ser plasmada como ideas puntuales dentro de las características que se busca con el sistema.

Estas fases pueden ser desglosadas en el siguiente diagrama, el cual es muy similar al del SMS.

Figura 7

Proceso de desarrollo para el SLR



Nota: Inspirado en (Moguel Márquez, 2018). Tomado de (Cobo Jaramillo, 2021).

Definir las preguntas de investigación. La principal motivación de este SLR es examinar las tecnologías, técnicas o herramientas más utilizadas para las temáticas de los *microservicios* y el *control telemático* ya que si bien el título del trabajo no lo refleja directamente: “*Diseño e implementación de un sistema domótico basado en la web of things (WoT) para la orquestación y descubrimiento de componentes cyber físicos.*” son dos puntos esenciales que tienen que analizarse más detalladamente, ya que la

domótica consiste en control telemático y la orquestación y descubrimiento son términos que están estrechamente ligados con lo que son los microservicios.

Tabla 4

Preguntas del proceso de investigación del SLR

Pregunta de investigación	Motivación
<i>¿Qué técnicas/tecnologías y herramientas existen para la orquestación y descubrimiento de componentes en sistemas domóticas?</i>	
Q1: ¿Qué técnicas, tecnologías y herramientas se utilizan para poder orquestar y descubrir componentes en sistemas domóticos?	Conocer qué tecnologías, técnicas o herramientas existen para poder orquestar, registrar y descubrir sistemas ciberfísicos en domótica.
Q2: ¿Qué tecnologías y/o técnicas son utilizadas para realizar <i>Benchmarking</i> de un sistema domótico?	Conocer qué tecnologías, técnicas o herramientas existen para realizar pruebas de rendimiento con el objetivo de medir y validar la funcionalidad de un sistema domótico.
<i>¿Qué técnicas/tecnologías existen para poder controlar sistemas ciber físicos en un entorno domótico orientado a la WoT?</i>	
Q3: ¿Qué técnicas y/o tecnologías son las más adecuadas para el control y la automatización de sistemas ciberfísicos en entornos domóticos orientados a la WoT?	Conocer qué tecnologías, técnicas o herramientas existen para controlar sistemas ciber físicos en un sistema domótico orientado a la WoT y cuál es su nivel de madurez.
Q4: ¿Cómo se puede garantizar una QoS adecuada en cuanto al control de un sistema domótico y su respuesta respecto a lo que requiera el usuario?	Conocer las técnicas, tecnologías y herramientas existentes para brindar y garantizar una QoS adecuada en un sistema domótico orientado a la WoT.

Nota: Se considera bastante oportuno investigar acerca de las pruebas de rendimiento o *Benchmarking*, debido a que esto es un requerimiento muy importante para poder evaluar el sistema en general y conocer cuáles son las limitantes del mismo.

Establecer los criterios de búsqueda primarios y secundarios.

Mucho de los criterios considerados para este SLR son similares a los del SMS, con la diferencia de que en el SLR se busca profundizar más en temáticas específicas y en los ámbitos técnicos que pueden englobar las mismas. Para la recopilación se usa las mismas librerías digitales que son: IEEE Xplore, MDPI, Science Direct y Springer Link, en estas se aborda únicamente los campos de estudio relacionados a la tecnología y también se da más importancia a conferencias importantes que fueron notables en el SMS, como son:

- Com-IT-Con.
- ICREST.
- STI.
- I-SMAC.

Definir cadena de búsqueda. Gracias a las preguntas planteadas en las etapas iniciales de este SLR, se puede determinar los términos importantes que será utilizados en la cadena de búsqueda.

Tabla 5

Términos definidos para la cadena de búsqueda del SLR.

Término principal	Términos alternativos
Orchestration & Discovery [Q1]	Orchestration OR Discovery
Microservices [Q1]	“Service oriented architecture” OR SOA OR “Microservice architecture”.
Benchmark. [Q2]	Baseline.
Control. [Q3]	Remote control.
QoS. [Q4]	Quality of service.

Por lo tanto, a partir de estos términos, la cadena de búsqueda que se define para recopilar la información es la siguiente:

```
("Orchestration & Discovery" OR Orchestration OR Discovery)
AND
("Microservices" OR "Service oriented architecture" OR SOA OR
"Microservice architecture".)
AND
("Benchmark" OR Baseline)
AND
("Control" OR "Remote control")
AND
(QoS OR "Quality of service").
```

Fijar los criterios de inclusión y exclusión. En primer lugar, únicamente se tiene en cuenta los artículos publicados entre enero del 2018 hasta septiembre del 2021, ya que se necesita la información más actualizada posible y de vanguardia, ya que esto llevará posteriormente a definir las características generales y necesarias que tendrá todo el sistema domótico. Además, para que cada uno de los documentos finales pueda ser considerado como relevante para el desarrollo del proyecto, debe cumplir con las siguientes características:

- Artículos que se encuentren completos, no solo resúmenes o reseñas.
- Que pertenezcan a las ramas de la tecnología pertinentes con este trabajo de investigación.
- Estudios en los que consten las técnicas, tecnologías o herramientas utilizadas para el diseño y la implementación de las temáticas propuestas, además preferentemente en las que consten las características o especificaciones técnicas de los dispositivos o componentes utilizados.

Determinar la estrategia de extracción de los datos. Al igual que en SMS, una vez se tiene toda la información y todos los documentos recopilados, es necesario sintetizar los datos con ciertas consideraciones, criterios e importancia que no se definen en este documento porque no influyen directamente en su desarrollo, simplemente se procede a realizar este proceso sin mayor inconveniente, de este modo se pudo registrar toda la información relevante, además se prioriza el enfoque que se le da a la investigación y finalmente se elimina los artículos repetidos y de poco interés para el trabajo.

Clasificación de los documentos extraídos. Las etapas del SLR son las mismas que se utilizan para el SMS, con la diferencia de que en esta sección se cuenta únicamente con una cadena de búsqueda:

1. La primera fase consiste en utilizar la cadena de búsqueda expuesta anteriormente y en los motores de búsqueda únicamente considerar: **Title + Keywords + Abstract**, de esta manera se obtuvo una cantidad total de 1544 resultados entre las diversas bibliotecas digitales, de la forma siguiente:
 - **IEEE Xplore:** 1081 resultados.
 - **Springer Link:** 39 resultados.
 - **ScienceDirect:** 116 resultados.
 - **MDPI:** 308 resultados.
2. La segunda fase consiste en utilizar la misma cadena de búsqueda, pero además aplicar los criterios de inclusión y exclusión, y finalmente en los motores de búsqueda

de cada librería utilizar: **Title + Keywords + Abstract**, obteniendo de este modo un total de 272 documentos entre las bibliotecas digitales, de la siguiente manera:

- **IEEE Xplore:** 102 resultados.
- **Springer Link:** 50 resultados.
- **ScienceDirect:** 38 resultados
- **MDPI:** 82 resultados.

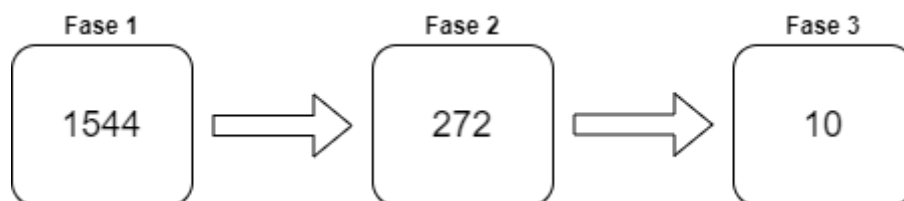
3. Para finalizar, la tercera fase, consiste en aplicar los mismos criterios utilizados en las fases anteriores y adicionalmente utilizar los motores de búsqueda de una manera más exhaustiva usando los filtros: **All Metadata**, en otras palabras, **Title + Keywords + Abstract + Full Article**, y además prestando mucha atención en el contenido importante de los artículos en sí, de este modo se ha obtenido un total de 10 documentos importantes y con relevancia para el trabajo, su distribución en las diferentes bibliotecas digitales es la siguiente:

- **IEEE Xplore:** 3 resultados.
- **Springer Link:** 2 resultados.
- **ScienceDirect:** 1 resultados
- **MDPI:** 4 resultados.

Adicionalmente en la figura 8, se puede ver el total de artículos que se extrajo en todo el proceso de este SLR:

Figura 8

Cantidad de artículos extraídos en todo el proceso del SLR.



Resultados. Después de aplicar todo el proceso de desarrollo del SLR, se pudo obtener un total de 10 artículos importantes y que aportan gran valor al desarrollo de este proyecto. Un enfoque que se pudo ver que es de gran importancia es el de la utilización de arquitecturas orientadas a servicios y arquitecturas de microservicios, ya que están estrechamente ligados a la orquestación y descubrimiento de los mismos que a su vez pueden estar relacionados a sistemas ciberfísicos. Otro de los puntos importantes se trata acerca de cómo controlar y automatizar este tipo de componentes, y todo consiste en el propósito que se quiera alcanzar, es así que todo depende de la programación interna del microcontrolador y las validaciones de los datos con respecto al entorno.

Conclusiones del SLR. Gracias a la recopilación de información de todo el proceso de este SLR, se puede proceder a responder las preguntas propuestas anteriormente, con las cuales se terminará de complementar el resto de información obtenida gracias al SMS y posteriormente poder definir a breves rasgos las características generales que debe cumplir el sistema domótico en su totalidad para alcanzar los objetivos propuestos en este trabajo.

Q1: ¿Qué técnicas, tecnologías y herramientas se utilizan para poder orquestar y descubrir componentes en sistemas domóticos?

Los términos de orquestación y descubrimiento están estrechamente ligados con conceptos como arquitectura orientada a servicios y arquitectura de microservicios, esto debido a que los microservicios van dirigidos a realizar una tarea en específico, en este caso irían relacionados a las acciones que pueden realizar los sistemas ciber físicos que se disponga en el sistema domótico, en todos los artículos obtenidos se mencionan arquitecturas de microservicios, dándoles un enfoque muy importante para poder alcanzar objetivos puntuales de cada sistema y sobre todo que sean escalables dependiendo de las necesidades del proyecto. Entre las herramientas que se mencionan están plataformas que utilizan lenguajes de programación como Java y frameworks como Spring, que están orientados específicamente a los microservicios (Kunold, 2019). Además, dentro de los patrones de diseño principales se encuentran arquitecturas RESTful, esto quiere decir que en el sistema se manejarán datos tipo JSON y el protocolo de comunicación HTTP, adicionalmente se debe tener en cuenta el manejo de APIs para que sean consumidos por los componentes que lo requieran (Kunold, 2019; Triboan, 2019). Es importante tener en cuenta los motores con los que funcionan los frameworks para poder saber la cantidad de datos que se puede gestionar, Springboot por ejemplo funciona bajo Apache Maven que puede recibir un máximo de 160 peticiones por segundo, antes de que el sistema colapse. (Zhang, 2020).

Q2: ¿Qué técnicas y/o tecnologías se utilizan para realizar Benchmarking de un sistema domótico?

Dentro del campo del Benchmarking y de pruebas de rendimiento, no se menciona mucho al respecto, pero en algunos artículos se mencionan prácticas importantes para determinar si un sistema puede garantizar la confiabilidad que se busca. En primer lugar, mencionan que es importante hacer validaciones en tiempo real y con entornos de prueba lo más reales posibles para saber cuáles son las fortalezas y debilidades del sistema (Chawla, 2021). También mencionan la importancia de ir testeando el proyecto por bloques y por ejemplo si se trata de herramientas de desarrollo basadas en Java, se puede utilizar Softwares como JUnit para hacer pruebas unitarias dentro de la aplicación (Zhang, 2020).

Finalmente para abordar el funcionamiento, la carga que pueden soportar y cuál es el rendimiento de todo el sistema en ciertas condiciones, se menciona un marco de prueba especializado en este tipo de pruebas llamado Gatling, en el cual se puede definir cualquier cantidad de condiciones para evaluar un sistema o una arquitectura, y así poder verificar cual es la cantidad máxima de peticiones por segundo que se podría manejar sin que exista ningún inconveniente con respecto a lo que el usuario perciba, ya que siempre es muy importante la calidad del servicio (Lyaskov, 2017).

Q3: ¿Qué técnicas y/o tecnologías son las más adecuadas para el control y la automatización de sistemas ciberfísicos en entornos domóticos orientados a la WoT?

De acuerdo a la síntesis elaborada de los artículos obtenidos, la tendencia actual en cuanto a automatizar los sistemas ciberfísicos, va orientada al aprendizaje de maquina y la inteligencia artificial, de este modo hay sistemas que se entrenan automáticamente a

sí mismos dependiendo de las variables que son específicas para cada entorno. Dentro de este campo de estudio como es la inteligencia artificial, se mencionan herramientas como OpenLight, un sistema de iluminación con autoaprendizaje y que se sitúa en el borde de la red (Bierzynski, 2019). Deep Neural Networks, soluciones en la nube (Chawla, 2021). Otro punto importante es la integración y el manejo de los datos utilizando protocolos específicos como HTTP, MQTT, XMPP, pero siempre se necesita un Gateway para poder recibir y enviar los datos, sin esto no se puede garantizar que el sistema sea completamente confiable (Agostinho, 2018).

En cuanto a los medios de transmisión de datos se mencionan algunos como Bluetooth, Zigbee, pero el más notorio y el que más impacto tiene en un sistema domótico es sin duda el WiFi, la facilidad de utilización de una red local brinda ventajas superlativas con respecto a otros tipos de transmisión (Kunold, 2019; Triboan, 2019).

Q4: ¿Cómo se puede garantizar una QoS adecuada en cuanto al control de un sistema domótico y su respuesta respecto a lo que requiera el usuario?

La calidad de servicio o QoS (Quality of Service) resulta importante en cualquier ámbito de la tecnología pero mucho más en entornos donde el usuario es el principal involucrado, como es el caso de una Smart Home, es por esto que es muy importante garantizar la transmisión de datos en el menor tiempo posible, y además el sistema debe saber manejar todo tipo de peticiones para las cuales está diseñado y posteriormente enviar una respuesta adecuada al usuario (Yue, Wu, & Xue, 2020); en otros artículos dan mayor prioridad a la experiencia y otros tienen mucho más marcado el uso de la retroalimentación para una garantía mayor se fundamentan mayormente en la experiencia del cliente como tal y para garantizar la calidad de servicio, algunos verifican los datos con un microservicio específico para esta labor y de ser correctos se

procede a la ejecución del requerimiento, caso opuesto se notifica al cliente que debería volver a llevar a cabo la acción, siendo de esta forma un proceso iterativo para cada cliente (Trunov, 2018; Badii, y otros, 2019).

Finalmente, algunos artículos mencionan que, para garantizar la calidad de servicio adecuada, se debe aplicar las métricas o variables específicas dependiendo de cada usuario, ya que, si esto llega a cumplirse, no habrá problema en cuanto a los tiempos de respuesta, eficiencia y rendimiento del sistema en el que se está trabajando y en este caso un sistema domótico (Zheng, Zhang, Zheng, Fu, & Liu, 2017; Laleh, Paquet, Mokhov, & Yan , 2018).

Otras conclusiones

Una vez se han respondido las preguntas propuestas anteriormente, también es importante mencionar algunos puntos importantes que influyen directamente en el desarrollo del trabajo:

- En primer lugar, es importante mencionar que con este SLR se ha podido complementar la información ya existente del SMS, de este modo se tiene todos los datos necesarios para el desarrollo de este trabajo.
- Se evidencio la importancia tanto del Benchmarking y de la calidad de servicio, esto en vista de que con un producto o servicio siempre se trata de cumplir con todas las expectativas del usuario y además garantizar que la transmisión de los datos siempre se cumpla sin ningún inconveniente.
- Si bien se pudo ver que existen algunas tecnologías para la transmisión de datos y para el control remoto de sistemas ciberfísicos o de componentes electrónicos, la que se utiliza mucho más es evidentemente el Internet y esto

se debe a la facilidad de usarla, ya que no requiere grandes conocimientos técnicos, más que saber cómo conectarse a una red local, por cable o WiFi.

- Todos los entornos orientados a la WoT se han popularizado en gran medida, ya que cada vez más se requiere que converjan el entorno físico y el entorno virtual, y es por ello que los servicios web se han convertido en un pilar fundamental en el desarrollo tecnológico.

Como conclusión final se tiene que los diseños e implementaciones de todos los artículos estudiados, son de propósito específico y orientados a objetivos propios de cada proyecto, sin embargo, es importante mencionar la gran versatilidad y escalabilidad que se puede lograr al tener este tipo de sistemas domóticos junto con la utilización de sistemas ciberfísicos y un medio de transmisión WiFi, esto definitivamente a ayudado a identificar el tipo de técnicas, tecnologías y herramientas que son las más adecuadas para cumplir con todo el proceso de este proyecto desde su diseño hasta su implementación.

Definición de las características del sistema

Una vez finalizado el SMS y SLR, se tiene recopilada la información necesaria para aclarar todas las ideas y se evidenció las múltiples técnicas, tecnologías y herramientas con respecto al diseño, implementación y desarrollo en cuanto al desarrollo de las arquitecturas de microservicios como en la implementación de sistemas ciber físicos y todos los componentes necesarios, por tanto es fundamental basar y conceptualizar cuáles son los instrumentos que se usaron y las características generales que el producto final posee.

Sistema domótico

Un sistema domótico debe ser capaz de automatizar una vivienda o cualquier edificación de acuerdo a los requerimientos de uno o varios usuarios, debe aportar una buena gestión energética, un alto nivel de seguridad y gran confiabilidad tanto en el bienestar y la comunicación en la red interior. Dentro de las características más comunes que debe brindar un sistema de este tipo, se encuentran:

- **Programación y gestión energética:** Este es un punto muy importante ya que, se debe manejar la climatización, las calderas de ser necesarias y las cortinas o persianas eléctricas de haberlas, entonces es muy importante distribuir muy bien la energía para evitar consumir en demasía.
- **Confort:** Esto conlleva en llevar a cabo las operaciones necesarias para que el usuario mejore su calidad de vida y sobre todo comodidad dentro de la vivienda en su vida diaria. Dentro de las tareas que puede tener disponibles están: el control de la iluminación, automatización de sistemas como cafeteras y el almacenamiento y gestión de la multimedia y de otros aspectos que puedan tener relacionado al internet.
- **Seguridad:** La seguridad en entornos como este, son una fortaleza, ya que, si se tiene bien implementados los protocolos o estándares necesarios, se tendrá un monitoreo constante de la vivienda, inclusive cuando el usuario se encuentre fuera de ella.
- **Accesibilidad:** Actualmente la mayoría de productos y servicios buscan ser inclusivos y accesibles para cualquier persona a pesar de tener discapacidades, y un sistema domótico no es la excepción, ya que, inclusive puede mejorar su calidad de vida en gran medida.

Microservicios

En la parte del Backend este enfoque arquitectónico, ayuda mucho en cuanto al desarrollo de software, ya que están relacionados y se comunican directamente con APIs que ya están bien definidas. Por los estudios realizados, las características más notables de este enfoque son las siguientes:

- **Son autónomos:** Cada microservicio individualmente se puede administrar y escalar sin afectar el funcionamiento de otros microservicios. Además, comparten ningún atributo o código con ningún otro componente de la arquitectura, y finalmente su comunicación se lleva a cabo de una API bien definida.
- **Son especializados:** Esto quiere decir que cada microservicio cumple con áreas específicas y está diseñado para cumplir con enfoques puntuales, además si en el caso de que escale mucho y crezca, puede ser dividido en varios microservicios diferentes, pero que cumplan con el mismo objetivo.
- **Escalables:** Cada microservicio puede ir adaptándose a las necesidades del negocio o del usuario, ya que su propósito es satisfacer todas las demandas que se presenten y medir con precisión si es conveniente ir creciendo hasta obtener ciertas características, esto con el fin de saber cuál es el costo total de implementación.
- Además de las características principales de los microservicios, también existen algunas otras como por ejemplos su agilidad en cuanto a diseño por que ocupan poco código, su fácil utilización ya que internamente se busca que no sean tan complejos, su tolerancia en cuanto a fallos ya que esto puede ser implementado con librerías específicas y finalmente la

ventaja que brindan de ser reutilizable y ser módulos pequeños que pueden ser adaptados a cualquier entorno o sistema dependiendo de los requerimientos

Después de los estudios realizados también se puede definir la herramienta de software más adecuada para implementar microservicios, la cual es *Spring Boot*, un framework de desarrollo open source basado en Java que está orientado, donde se puede diseñar una arquitectura de microservicios, donde cada uno estará encargado de guiar y dirigir cada uno de los procesos necesarios.

Sistema ciber físico

Uno de estos sistemas, a diferencia de uno tradicional, tiene como característica principal el estar diseñado de tal manera que pueda conectarse con otros elementos de una red, donde pueden interactúan entre sí. Esto se logra con algunas herramientas y tecnologías de acuerdo a la aplicación que se quiera dar. En este trabajo se utiliza como controlador una pequeña computadora que, por sus características, su flexibilidad y su versatilidad puede aceptar sistemas operativos diferentes y algunos lenguajes de programación, además también puede estar acoplada sin ningún problema al Backend del sistema que está desarrollado con Spring Boot, finalmente por sus múltiples puertos de entrada y salida, permite integrar la cantidad necesaria de sensores y actuadores para lograr los objetivos propuestos.

Control y Automatización

No quedan dudas de que el Internet es la mejor tecnología para poder controlar y automatizar todos los componentes del sistema domótico propuesto. El papel de los dispositivos móviles también es muy importante ya que al estar al alcance de la mano brindan una respuesta muy rápida si algo pudiese acontecer dentro del sistema.

Finalmente se encuentran las tendencias y los protocolos de comunicación más utilizados, siendo los que funcionan bajo TCP/IP, entre ellos HTTP que permite que los datos se transmitan de una manera rápida, adecuada y que están muy bien adaptados a funcionar en arquitecturas de microservicios donde los servicios web son de tipo RESTful y se maneja datos JSON tanto para la comunicación interna de todo el software específico desarrollado, como también en el código orientado al manejo control y automatización de los sistemas ciber físicos.

Capítulo II

Tecnologías Relacionadas

Domótica

El eje troncal del trabajo es la domótica, este término va relacionado con todo el conjunto de tecnologías aplicadas al control y automatización de procesos dentro de una vivienda, convirtiéndola en inteligente ya que, procesa todas las ordenes que se le dan al núcleo del sistema y de este modo se puede gestionar los datos que entrega alguno de los sensores y también hacer que alguno de sus actuadores realice la tarea para la que está enfocada (Asociación Española de Domótica e Inmótica, 2019).

Características de la Domótica

Entre las características más importantes que brinda la domótica se encuentran:

- **Ahorro energético:** Gestiona de manera inteligente la iluminación, climatización, servicios de agua caliente, electrodomésticos y todos los dispositivos tecnológicos actuales.
- **Accesibilidad:** Facilita que cualquier persona pueda manejar cualquier entorno dentro del sistema domótico, independientemente de si tenga o no algún tipo de discapacidad.
- **Seguridad:** Brinda mayor confiabilidad respecto a la vigilancia del hogar, ya que está orientado al cuidado de las personas del interior de la vivienda y al ser algo automatizado e inteligente, evita cualquier tipo de intrusión (Asociación Española de Domótica e Inmótica, 2019).

Ventajas de la Domótica

Entre las ventajas que brinda la utilización e implementación de sistemas domóticos, se puede mencionar varios entornos y escenarios en los cuales cumple un papel importante:

- **Comunicación:** Facilita la transmisión de cualquier archivo multimedia mediante las redes locales que comparten el acceso a internet.
- **Mantenimiento:** Tiene gran capacidad de incorporar un sistema de gestión y mantenimiento telemático.
- **Ocio:** Permite que la fácil utilización de los dispositivos tecnológicos actuales, incluyendo las consolas y plataformas de videojuegos.
- **Salud:** Si se tiene una buena centralización en cuanto al monitoreo de las variables de salud individuales se puede facilitar la rápida respuesta de las entidades de salud en caso de alguna emergencia.
- **Compras:** La completa automatización de una vivienda, inclusive permite que se pueda registrar el inventario de una alacena o de algún otro insumo, esto permite que si llegase a faltar, se pueda realizar una compra automáticamente (Asociación Española de Domótica e Inmótica, 2019).

Desventajas de la Domótica

Entre las ventajas más importantes de la implementación de un sistema domótico, más que de utilización, son respecto a la implementación y a las instalaciones iniciales, y son:

- **Costo inicial:** El precio puede llegar a ser muy elevado y es muy importante para seguir con el desarrollo de todo el sistema, ya que se debe cubrir toda la vivienda en su totalidad.

- **Refacción:** Si se llegase a dar algún tipo de falla o si se dañase algún componente de la infraestructura, esto puede llegar a ser muy tedioso, ya que tocaría identificar el motivo de la falla y esto representa algo muy complejo y costoso.
- **Transmisión de datos:** Si se tiene muchos usuarios dentro de la red de la vivienda, puede darse un escenario de cuello de botella en cuanto a la transmisión de los datos, ya que no está preparada para gestionar volúmenes de datos masivos.
- **Tipo de conexión:** Este es un factor muy importante a tener en cuenta, ya que si no hay otra posibilidad aparte de hacer una conexión de anillo esto puede llevar a que toda la infraestructura se complique y sea muy difícil de implementar (CEAC, 2021).

Inmótica

Este término es homónimo al de domótica, sin embargo, en este caso se trata del control y automatización de todo tipo de edificios que no estén orientados a viviendas, entonces pueden encontrarse hoteles, hospitales, centros educativos y todo edificio terciario que pueda estar relacionado con el servicio a las personas (Asociación Española de Domótica e Inmótica, 2020).

Características de la Inmótica

Entre las características más importantes de la inmótica se encuentran ciertas funciones que pueden regularse dependiendo de la necesidad de cada entorno, y son:

- **Regulación automática:** Esto va orientado a dispositivos, herramientas o piezas móviles dentro del edificio que pueden ser reguladas, como, por

ejemplo: la calefacción, las puertas, la ventilación, la iluminación y las persianas.

- **Control centralizado del edificio:** Esto puede ser utilizado para programar acciones específicas dependiendo de las actividades que desarrollen los trabajadores del edificio, como por ejemplo determinar horarios donde la luminaria funcione en su totalidad o parcialmente. Además, se puede optimizar la centralización de todo el sistema, ya que los dispositivos irían conectados a reguladores o puntos de consigna comunes.
- **Gestión técnica de energía:** Esto sobre todo va orientado a la detección de fallos ya sea dentro de los edificios o en alguno de sus sistemas técnicos que permiten su funcionamiento, además facilita que se pueda presentar la información de consumo energético dividido por sectores y así se pueda proponer posibilidades de mejora (Asociación Española de Domótica e Inmótica, 2020).

Ventajas de la Inmótica

Las ventajas de la inmótica son muy similares a las ventajas de la domótica, la diferencia radica en la magnitud de cada una, ya que automatizar un edificio completo es mucho más complejo que automatizar solo una vivienda. Entonces el enfoque que se les da a dichas ventajas es el siguiente:

- **Seguridad:** Se puede realizar la detección de gases peligrosos, intrusión en las instalaciones, fallas técnicas del edificio, se tiene mayor certeza en horarios de funcionamiento.

- **Confort:** Debido a la automatización de todos los sistemas del edificio, esto brinda mucha mayor comodidad para todas las personas que se encuentren en el edificio, además si no se da ningún error, un sistema inmótico puede funcionar varios años sin problemas y sin ningún mantenimiento.
- **Ahorro y eficiencia energética:** En industrias o edificios muy grandes este es un punto crucial ya que, si se tiene una optimización y una gestión inteligente respecto a la utilización de la energía eléctrica, esto se refleja en un gran ahorro para la entidad dueña de dicho edificio y de este modo se puede aprovechar mejor los recursos.
- **Comunicación:** Este punto es muy importante sobre todo en edificios como bibliotecas o instituciones educativas, ya que un fácil acceso a la información desde cualquier lugar del edificio o del campus, mejora en gran medida la experiencia de los usuarios y de las personas que utilizan la infraestructura del edificio (Sainel, 2019).

Desventajas de la Inmótica

Las desventajas de la inmótica aparte de los altísimos costos debido a su magnitud, y del tamaño de la infraestructura de los edificios, van orientadas a las vulnerabilidades que pueden presentar, y son:

- Sistema que depende mucho del internet en toda su infraestructura.
- Es muy vulnerable a ataques digitales ya que, al estar conectado al internet en toda su infraestructura, esto hace que un virus pueda ser introducido más fácilmente.

- Pueden darse fallas de sobrecarga debido a la gran cantidad de datos que pueden recorrer dentro de la red.
- Si hay alguna intrusión al sistema, puede convertirse en algo sumamente inseguro y atentar contra la intimidad de las personas (Sainel, 2019).

Arquitecturas de servicios

Un punto importante dentro del trabajo son las arquitecturas orientadas a servicios, con el fin de poder gestionar los datos de tal manera que se adapten a los objetivos del sistema domótico propuesto. Es así que se pueden nombrar arquitecturas como SOA o una arquitectura de microservicios, y todas las herramientas y funcionalidades que estas pueden ofrecer.

SOA (*Service Oriented Architecture*)

Esta arquitectura, así como su nombre lo indica, está orientada a ocupar servicios de cualquier tipo como su núcleo para poder cumplir procesos, rutinas o secuencias específicas para alcanzar con los objetivos o requerimientos de algún negocio o empresa, además de facilitar la interacción con los componentes tanto internos como de terceros (eCityclic, 2020). En la figura 9 se aprecia como SOA facilita la integración de los múltiples sistemas que componen un modelo de negocio o empresa.

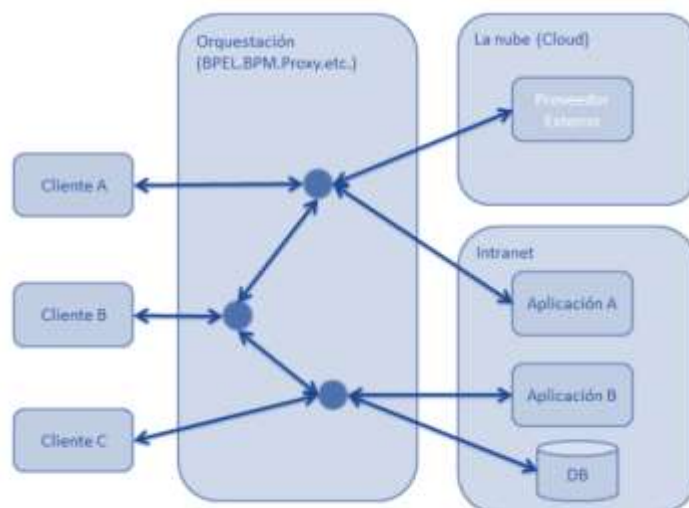
Características de SOA. Las características principales de SOA van orientadas a su funcionalidad y son las siguientes:

- Se puede reutilizar sus componentes.
- Se puede integrar un proceso a múltiples entornos de desarrollo.

- Es muy ágil en cuanto a su desarrollo, ya que se lo puede hacer en un tiempo corto.
- Todos los servicios deben tener características únicas que los identifiquen.
- Por último, SOA no es independiente en su totalidad ya que, o bien puede depender de múltiples servicios o bien puede ser un servicio único para un sistema mucho más grande (eCityclíc, 2020).

Figura 9

SOA en un modelo de negocio



Nota: Tomado de (Blancarte, Oscar Blancarte Software Architect, 2014).

Ventajas de SOA. Las ventajas de SOA se ven reflejadas en un entorno donde se tenga ya muy bien definido el modelo de negocio o de empresa y son las siguientes:

- La eficiencia aumenta en todos los procesos.

- Disminuye todos los costos de inversión que pueden darse en el sistema.
- Los costos de mantenimiento son muy bajos.
- Si se trabaja con una herencia o relación entre productos o servicios, se facilita la implementación.
- Incentiva la innovación en todos los campos, además que para un negocio esto representa estar a la vanguardia con el mercado.
- Simplifica todo el diseño, implementación y despliegue de una arquitectura. (eCityclic, 2020).

Desventajas de SOA. Así como existen ventaja, las desventajas de SOA, también son muy evidentes y entre algunas se encuentran las siguientes:

- Requiere que se tenga mucho cuidado con respecto al enfoque que se le va a dar, esto con el fin de que se mantenga una coherencia desde el principio al final del modelo de negocio.
- Siempre se debe tener en cuenta que los servicios son simples, y cumplen con una tarea específica, es por ello que no deben ser complejos.
- El desarrollo de la arquitectura puede ser extenso ya que está compuesta por muchos servicios.
- Siempre se debe tener una planeación y orientar a la arquitectura a que sea escalable.
- La falla de un servicio puede ocasionar que toda la arquitectura falle.
- Si un servicio dependiente de otros llega a cambiar, invalidaría el flujo del proceso presentes y la consistencia de todos los datos (eCityclic, 2020).

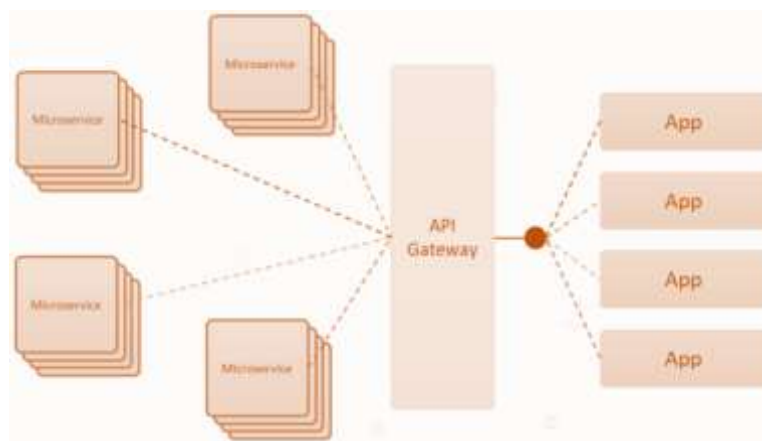
Arquitectura de Microservicios

En los últimos años las arquitecturas de microservicios se han tornado en las más populares y en un pilar fundamental para las arquitecturas de grandes empresas como Amazon, Netflix, etc. Su funcionamiento consiste en crear pequeños servicios o componentes, que están orientados en realizar tareas específicas, y que tienen la característica de ser independientes de los otros componentes de la aplicación.

Una arquitectura de microservicios busca dividir toda su funcionalidad en pequeños servicios que puedan ser autosuficientes en todos los aspectos, su diferencia con las arquitecturas monolíticas que también son ampliamente utilizadas es que, no todo funciona en conjunto y si llegase a fallar un componente, esto no ocasionará que se dé un fallo en cascada, además los microservicios tienen la peculiaridad de funcionar a través de un componente llamado API Gateway, el que permite el registro, acceso y control de todo el resto de microservicios. En la figura 10 se puede observar de manera general una arquitectura de microservicios (Blancarte, Oscar Blancarte Software Architect, 2018).

Figura 10

Diagrama general de una arquitectura de microservicios



Nota: Tomado de **(Blancarte, Oscar Blancarte Software Architect, 2018)**.

Características de una arquitectura de microservicios. Entre las características de una arquitectura de microservicios se encuentran las siguientes:

- Los microservicios deben poder comunicarse entre sí, para poder cumplir con los objetivos del producto final.
- Cada microservicio puede operar independientemente bajo su formato y estilo, además pueden diferenciarse inclusive en el lenguaje de programación y base de datos.
- El despliegue de los microservicios no necesariamente debe ser en la misma infraestructura, su característica de desacople permite que funcionen desde cualquier servidor y lugar.
- Siempre deben funcionar bajo algún patrón de diseño específico, por lo general suele ser REST.
- El mantenimiento a microservicios específicos no afecta al resto que estén al mismo tiempo en funcionamiento. (Blancarte, Oscar Blancarte Software Architect, 2018).

Ventajas de una arquitectura de microservicios. Los microservicios brindan muchas ventajas con respecto a otros tipos de arquitecturas, entre estas se encuentran:

- Los microservicios son altamente escalables, permitiendo implementar varias instancias del mismo y utilizar balanceo de cargas entre estas.
- Su implementación es ágil debido a su sencillez de desarrollo y la independencia de unos con otros.
- Las pruebas que se realiza en los mismos son muy intuitivas y permiten un despliegue fácil, ya que no dependen de fuentes externas.
- Pueden tener un gran alcance si trabajan en conjunto, esto dependerá del propósito del producto final.

- Son reusables e interoperables, esto quiere decir que se puede reutilizar su código e inclusive implementarlos en un lenguaje de programación diferente sin afectar el objetivo final (Blancarte, Oscar Blancarte Software Architect, 2018).’

Desventajas de una arquitectura de microservicios. Las desventajas de los microservicios quizá no sean muy evidentes, pero existen, entre ellas se tienen:

- La característica de ser una arquitectura distribuida trae consigo la desventaja de que se agrega latencias significativas si se tiene múltiples servidores en distintos lugares, esto ya que el tiempo de respuesta evidentemente aumentará.
- Los puntos de falla aumentan ya que se tiene muchos componentes para ser gestionados.
- Su funcionamiento no es lineal, ya que la trazabilidad de la aplicación o puede tener un seguimiento ordenado, siempre se debe tratar a la arquitectura por bloques específicos orientados a sus propias tareas.
- Se requiere de un equipo capacitado y que conozca el funcionamiento de cada una de las funcionalidades que puede tener un microservicio, esto con el fin de que sea una arquitectura robusta y que tenga las tecnologías relacionadas correctamente administradas.
- Finalmente, una vez desplegada la arquitectura, se debe tratar de que el consumo de los recursos sea óptimo, caso contrario puede representar costos muy altos (Blancarte, Oscar Blancarte Software Architect, 2018).

REST (Representational State Transfer)

REST se ha tornado muy relevante actualmente debido a su fluida interacción con los sistemas distribuidos actuales en la World Wide Web, o más claro en el Internet.

Este estilo o patrón de diseño permite manejar cantidades masivas de datos de diferentes tipos , ya que, su protocolo de comunicación por defecto es HTTP, el cual es muy ligero tanto para ejecutarlo, obtener, enviar y procesar información. Por lo general REST utiliza mensajes de tipo JSON. (Wikipedia, 2020)

REST facilita el desarrollo de una API REST utilizando los métodos de HTTP propios que son GET, POST, PUT y DELETE, esto permite que se puedan consumir y ocupar las funcionalidades de la arquitectura, donde finalmente se almacena los datos como URLs que pueden limitarse por el desarrollador dependiendo del propósito del BACKEND. (Rosa Moncayo, 2018)

Características de REST. Las características de REST se hacen evidentes cuando se consume una API REST, donde se puede distinguir en gran medida la utilización de HTTP y se puede definir las siguientes propiedades:

- Las peticiones HTTP funcionan como un REQUEST a un servidor, el cual gestiona los datos y los prepara de manera adecuada para ser enviados de vuelta como un RESPONSE.
- HTTP no necesita modificaciones o invenciones nuevas, se encuentra muy bien estandarizado.
- Los métodos HTTP permiten cumplir con la mayoría de requerimientos que se puede tener con respecto a los datos:
 - DELETE: Borrar un recurso o un dato.
 - GET: Obtener recurso en concreto.
 - PATCH: Modificar un recurso que no sea un dato.
 - POST: Crear nuevos recursos.
 - PUT: Modificar recursos y datos.

- EL manejo de objetos mediante URLs facilita el envío y recepción de datos, sobre todo cuando se quiera obtener recursos muy específicos (Rosa Moncayo, 2018).

Ventajas de REST. Al tratarse de un patrón de diseño ya conocido, las ventajas ya son bien conocidas y son las siguientes:

- Permite desacoplar el cliente del servidor, permitiendo que puedan ser desarrollados en lenguajes de programación diferentes y no necesariamente deban seguir paradigmas específicos.
- Los diseños si se tratan de servicios van orientados a dominios o direcciones específicas, aumentando la posibilidad de maneja los datos en formato JSON.
- Es independiente de cualquier sistema operativo o plataforma de desarrollo.
- Es muy fácil valorar su rendimiento por el desacople de cliente y servidor, entonces pueden ser valorada en sus diferentes entornos, lo que también brinda mayor escalabilidad a cualquier aplicación.
- Es sumamente ligero por su protocolo por defecto que es HTTP (Rosa Moncayo, 2018; Chakray, 2020).

Desventajas de REST. Las desventajas de REST también son muy evidentes entre los desarrolladores que utilizan este patrón de diseño:

- No existe un estándar específico del uso de los métodos HTTP, entonces la utilización de cada uno puede ser escogida por cada desarrollador.

- API REST no necesariamente tiene contratos cliente – servidor por defecto, esto hace que pueda tornarse menos segura.
- Uno de los grandes inconvenientes de REST es el no poder anejar cantidades masivas de datos por su característica de una sola llamada y respuesta a la vez, sin embargo, esto puede verse solventado en cierta medida si se utiliza RESTful (Chakray, 2020).

Tecnologías de Back-End

Cuando se habla de Backend se hace referencia a toda la parte de la arquitectura de software que se encarga de la lógica específica de implementación o de negocio. Todas estas tareas son imperceptibles pero fundamentales para el funcionamiento correcto de toda la arquitectura como por ejemplo la comunicación con el servidor o él envió de información a la base de datos.

En el Backend se debe ser cuidadoso y muy meticuloso, ya que un error de cualquier tipo puede ocasionar que toda la aplicación falle. Las funciones del Backend pueden ser brevemente mencionadas y son:

- Optimización de procesos con funciones especializadas.
- Manejo, gestión y almacenamiento de la información en bases de datos.
- Funcionalidades específicas que pueden realizarse utilizando módulos o librerías especiales.

En el Backend se puede utilizar diferentes lenguajes de programación y tecnologías, por lo que es muy necesario que cualquier profesional que se dedique a este campo debe estar estrechamente familiarizado con lo que pueda llegar a utilizar. (Arjonilla, 2017). En el Backend se tiene lenguajes muy utilizados como PHP (Hipertext Preprocessor) orientado a aplicaciones web y especialmente popular ya que puede ser

introducido en HTML. (The PHP Group, 2020). Ruby que es un lenguaje balanceado, fue creado a partir de muchos otros lenguajes generando un lenguaje funcional e imperativo y altamente flexible (Ruby, 2020). Python que es un lenguaje de programación muy popular actualmente por su código simple y a su sintaxis que es muy fácil de entender (Wikipedia, 2021). Pero a continuación se detalla los lenguajes que tienen un impacto importante para el trabajo:

JAVA

Es un lenguaje de programación de alto nivel, y debido a sus tecnologías de desarrollo, está altamente relacionado con el diseño, implementación y despliegue de aplicaciones Web, donde se busca que sean más útiles e interesantes. Es necesario poner en claro que no es lo mismo que *JavaScript*, una tecnología que resulta más sencilla y exclusivamente para exploradores Web.

Java al ser orientado a objetos, facilita a los usuarios realizar muchas tareas cotidianas. Java puede encontrarse en muchos productos y servicios actuales, como juegos online, aplicaciones de mensajería, aplicaciones fotográficas, etc.

Finalmente, es muy importante mencionar que Java es el pilar fundamental de casi todas las aplicaciones de red, además está fuertemente estandarizado en cuanto a la compilación y distribución de muchas aplicaciones móviles, juegos, servicios web, aplicaciones empresariales, etc. Dentro de lo que se puede realizar utilizando Java se encuentra:

- Permite virtualización en plataformas diferentes a donde se desarrolló.
- Funciona muy bien con aplicaciones web donde domina HTML.
- Permiten gran nivel de personalización en aplicaciones ad-hoc.

- Una vez se compila y distribuye una aplicación desarrollada en Java, esta puede adecuarse y funcionar en cualquier tipo de dispositivo electrónico, como celulares, sensores, módulos WiFi, etc. (Oracle, 2020).

Processing

Este lenguaje de programación de alto nivel, es similar a C++. Es un lenguaje de propósito general y este asociado a un sistema operativo de tipo UNIX. Sus características específicas son la siguientes:

- Trata de manera similar a un lenguaje de nivel medio todo lo que son caracteres, números, bits, direcciones de memoria, entre otros.
- Su portabilidad ayuda a que se lo pueda utilizar en sistemas compiladores, intérpretes y editores textuales.
- Es fácil comprenderlo y profundizar en este lenguaje por su sintaxis y forma de sus algoritmos.

Finalmente, una de las grandes ventajas de utilizar este lenguaje y sobre todo Arduino. es que, por su popularidad, toda la comunidad ha desarrollado una gran cantidad de librerías que pueden ser utilizadas para cumplir con propósitos específicos, así como la conexión a redes, la conexión a un Broker, como el utilizado en el trabajo y claramente la importancia de poder manejar sensores y actuadores electrónicos (BeJob, 2017).

MQTT

Este no es un lenguaje de programación, si no un protocolo de comunicación, sus siglas significan MQ Telemetry Transport, funciona con un estándar M2M, es decir de máquina a máquina y es de tipo message queue, recibe mensajes que se van

almacenando en una pila y van siendo enviados dependiendo de la velocidad de respuesta de un sistema. Se basa en TCP/IP y cada conexión es reutilizada en cada comunicación. Sus características son las siguientes:

- Usa mensajería push, con patrón publicador y suscriptor.
- Los mensajes se disponen en tópicos organizados jerárquicamente.
- Cada nuevo cliente utiliza un mensaje CONECT y recibe un mensaje CONNACK.
- Para suscribirse a un tópico se usa el mensaje SUSCRIBE y el cliente recibe un mensaje SUBACK.
- Finalmente, para publicar se utiliza el mensaje PUBLISH.

Como conclusión, MQTT se ha afianzado como uno de los protocolos y estándares para aplicaciones IoT por excelencia, ya que no solo es simple de utilizar, sino que brinda una amplia gama de funcionalidades y características que brindan seguridad, y confiabilidad, además se puede profundizar mucho más en su aplicabilidad y funcionamiento (Navarro, 2021).

Tecnologías de Front-End

En el desarrollo de Software, cuando se hace referencia al frontend se trata todo lo contrario con respecto al backend. El frontend en palabras cortas, es todo lo que el usuario puede ver o percibir, donde se encuentran todos los elementos gráficos y de la aplicación, los cuales permiten la interactividad con el usuario, dichos elementos pueden tener características muy específicas y especiales, se les puede incluir estilos, colores y animaciones. Al igual que el Backend, el desarrollo del Frontend se lo realiza usando lenguajes de programación específicos y más cercanos a la interacción y comprensión de las personas que vayan a ocupar las aplicaciones o interfaces

(Nestrategia, 2021). Así como para el Backend, los profesionales que trabajen en Frontend deben estar estrechamente relacionados con las tecnologías y las prácticas más utilizadas en este campo.

HTML

Este es un lenguaje de marcado, funciona con etiquetas que permiten definir la estructura de todo el contenido que se desea en una aplicación Web. Cada etiqueta tiene las siguientes características.:

- Apertura de la etiqueta.
- Contenido de la etiqueta.
- Cierre de la etiqueta,

Se deben definir los atributos de una etiqueta para que estas funcionen de manera adecuada. Finalmente, lo más importante que hay que tener en cuenta es la semántica, es decir el orden general de las etiquetas adecuadas y correspondientes para cada sección (Universidad de Murcia, 2019).

CSS

CSS es el encargado de la personalización de una aplicación web, su función es dar estilo a las etiquetas HTML previamente estructuradas. Con CSS se puede dar color, tamaño, posición y algunas otras características que pueden tener los elementos de una aplicación Web. Presenta características primordiales:

- Escribir la etiqueta con llave de apertura.
- Escribir el nombre de lo que se quiere modificar.
- Escribir el nuevo valor que se quiere dar.
- Finalmente, una llave de cierre.

Saber CSS resulta de gran importancia para ser un desarrollador, ya que ayuda a seguir buenas prácticas y elaborar código propio para ahorrar dinero y tiempo (Universidad de Murcia, 2019).

JavaScript

JavaScript es un lenguaje de programación multiparadigma dinámico e imperativo, es orientado a objetos. Tiene dos funciones principales, la primera es interactuar con los usuarios y la segunda es que sirve para comunicarse con el Backend, sirve como enlace entre las acciones de los usuarios en la aplicación Web y el Gateway, Proxy o API para poder responder adecuadamente a los requerimientos que se solicite. Es un puente que funciona para recibir y enviar peticiones, además de manejar los datos que se quiere utilizar para algún propósito en específico (Universidad de Murcia, 2019).

Agente de orquestación y descubrimiento

Orquestación y descubrimiento de servicios

Estos son conceptos claves dentro del trabajo, ya que, si bien se busca utilizar servicios, estos deben relacionarse directamente con los sistemas ciberfísicos a implementar. Debido a esto, los servicios o microservicios deben estar publicados, etiquetados correctamente y ser accesibles desde los metadatos de la red. Entonces se debe tener en cuenta términos clave como: La Gobernanza, para garantizar optimizar la reutilización y funcionamiento de los servicios publicados, además de obtener un crecimiento ordenado de los recursos y un mantenimiento oportuno conforme a las necesidades (Morales M. J., 2021).

Hay estándares que permiten la implementación de este principio, y dentro de estos se encuentra: *La descripción, el descubrimiento y la integración universal (UDDI)*

que registra todos los servicios en un documento XML. Todos los registros pueden obtenerse o consumirse a través de APIs o GUIs especializadas, y también hay mecanismos como WS-Discovery que facilita las acciones en registros centralizados (Alzaghoul, 2021).

Oracle dispone de ciertos contenedores ya se para J2EE o OC4J y funcionan con UDDI, el cual tiene una guía bien definida, donde se menciona tópicos esenciales para su correcta implementación:

- **Registro UDDI:** Orientado a tres tipos de búsqueda:
 - *Páginas en blanco:* Información de contacto.
 - *Páginas amarillas temáticas:* Información de datos industriales estandarizados.
 - *Páginas verdes de servicios:* Información de los servicios web que son publicados por su proveedor.
- **Archivo UDDI:** Basado en estructuras de datos específicas para cada uno de los servicios publicados:
 - *businessEntity:* Información descriptiva del negocio que publica el servicio.
 - *businessService:* información descriptiva del área técnica a la que se enfoca el servicio web.
 - *bindingTemplate:* Información técnica del servicio web y referentes a su interfaz.
 - *tModel:* Descripción de los servicios web orientados a facilitar la búsqueda dentro de todos los servicios registrados.

Además, Oracle también dispone de construcciones empresariales donde el manejo de los datos es más extenso y se usan tecnologías diferentes como *Oracle UDDI Enterprise Web Service Registry* (Oracle, 2012).

Seguridad

La seguridad es algo de vital importancia para cualquier producto o servicios tecnológico, esto debido a las vulnerabilidades que se han ido presentado a la par con la innovación, esta disciplina busca proteger todos los datos que pueden estar almacenados en un sistema, con el fin de garantizar la integridad y privacidad del mismo. La seguridad puede darse en cualquier entorno, ya sea de software, hardware, o en la red en general, el objetivo es el mismo, de brindar una resistencia adecuada y proactiva frente a cualquier ataque (ITTGWEB, 2021).

Como se mencionó la seguridad puede darse en cualquier entorno, y un producto o servicio siempre debe estar regido por la misma, ya que es la seguridad de su información y también el prestigio de las empresas distribuidoras u ofertantes de los mismos, están orientadas a estos tres tipos de seguridad:

- *Seguridad de hardware*: Seguridad robusta en un dispositivo físico como: firewalls o módulos de seguridad de hardware (HSM). También debe tenerse en cuenta la seguridad contra daños.
- *Seguridad de Software*: Su objetivo es salvaguardar la aplicación o software de cualquier tipo de ataque malicioso ataques maliciosos, sin que existan interrupciones en la presencia de cualquier riesgo. Hay programas especializados como: antivirus, antimalware, cortafuegos, antispam, y otros más de control que están enfocados a peligros específicos.

- *Seguridad de red*: Su objetivo es proteger la red, está presente dentro de la comunicación de extremo a extremo, salvaguardando los protocolos y la información, evita que circule cualquier amenaza o virus por la misma. Los programas específicos para este tipo de seguridad son los: antispyware, los sistemas de prevención de intrusiones (IPS) y las redes privadas virtuales (VPN) (Universidad Internacional de Valencia, 2018).

Herramientas de Software

SpringBoot

SpringBoot es un framework de desarrollo diseñado para facilitar el diseño e implementación de aplicaciones escritas en Java y Spring, esto ya que configurar manualmente Spring es muy difícil por sus múltiples archivos XML y la meticulosidad que requieren. SpringBoot proporciona un kit de herramientas de desarrollo que tienen el objetivo de desarrollarlas una sola vez y después solo ejecutarlas, su sencillez radica en estar basado en anotaciones y no archivos XML completos (Perry, 2017). Las características que han popularizado SpringBoot y es ampliamente utilizado son las siguientes:

- Es intuitivo, porque sus valores son predeterminados (SpringBoot, 2021; Perry, 2017).
- Es personalizable, ya que permite cambiar todos los valores predeterminados, además esto es posible tanto inicialmente, como en el desarrollo (Perry, 2017). Además, todas librerías externas se pueden revisar, testear y manipular (SpringBoot, 2021).

- Su inicialización depende del tipo de aplicación que se va a desarrollar, además todos los inicializadores tienen soporte para muchos sistemas operativos como Linux, Mac y Windows (Perry, 2017).

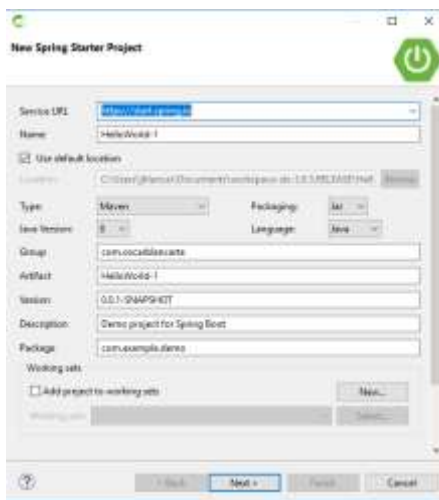
Para generar una aplicación basada en Spring, por lo general se utiliza Spring Suite IDE, que se puede ver en la figura 11. Solo se necesita definir el tipo de proyecto (Maven, Gradle), el lenguaje de programación (Java, Kotlin, Grovy) y la dependencia (Web por defecto).

Después de generar un proyecto, se genera un prototipo de la clase Main de aplicación, este es el Core en el caso de que se quisiera correr una aplicación más grande. Es importante mencionar que si lo que se desea es un servicio web se debe habilitar a la clase como REST, para que pueda ser testada en un navegador local.

Finalmente, en un Tomcat embebido se despliega el proyecto y para visualizarlo en el localhost se puede utilizar un navegador y se podrá observar algo similar a la figura 12.

Figura 11

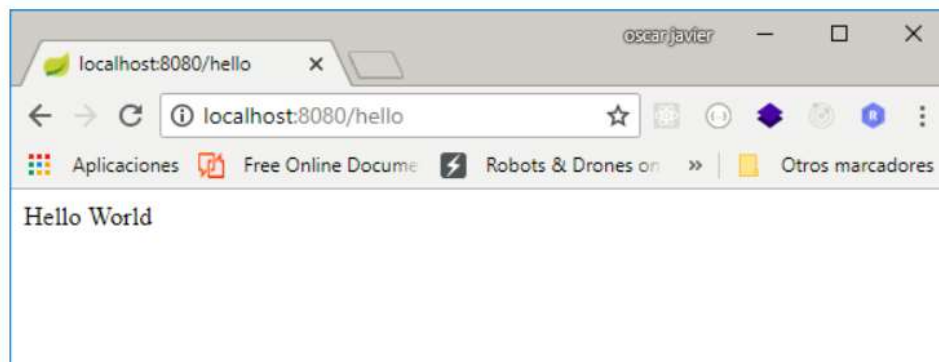
Spring Boot IDE



Nota: Figura tomada de (Blancarte, Oscar Blancarte Software Architect, 2018).

Figura 12

Navegador redireccionado al localhost



Nota: Figura tomada de **(Blancarte, Oscar Blancarte Software Architect, 2018)**.

Hashicorp Consul

Hashicorp desarrolló Consul para facilitar la comunicación entre cualquier servicio, ya sea consumidor o productor. La ventaja y el enfoque que se le da en el trabajo es para el registro, orquestación y descubrimiento de los servicios que estarán relacionados directamente con los sistemas ciberfísicos que constarán en el sistema domótico. Las principales características de Consul son las siguientes:

- Permite registrar, descubrir y orquestar servicios.
- A nivel de servicio o de nodo consúl clúster permite realizar un chequeo de calidad.
- Permite almacenar los datos utilizando claves y valores, esto centraliza la configuración.
- Distribución de data centers, esto permite que todo funcione correctamente sin la necesidad de incluir capas de configuración (Paradigma Digital, 2018).

Cónsul Agent. El concepto fundamental bajo el que se maneja Consul es el de Agente, esto es un proceso de larga duración que se ejecuta en cada nodo que compone a un sistema de Consul, su responsabilidad es recopilar o entregar información acerca de los mismos. Se ejecuta utilizando el protocolo HTTP e integra un DNS que permite que se pueda desplegar como cliente o como servidor (Paradigma Digital, 2018).

Agente Servidor. Se encarga de almacenar y replicar cualquier dato del sistema y responder peticiones, sin importar si las peticiones son propias o de un datacenter externo, entonces el servidor se encarga de la comunicación entre nodos.

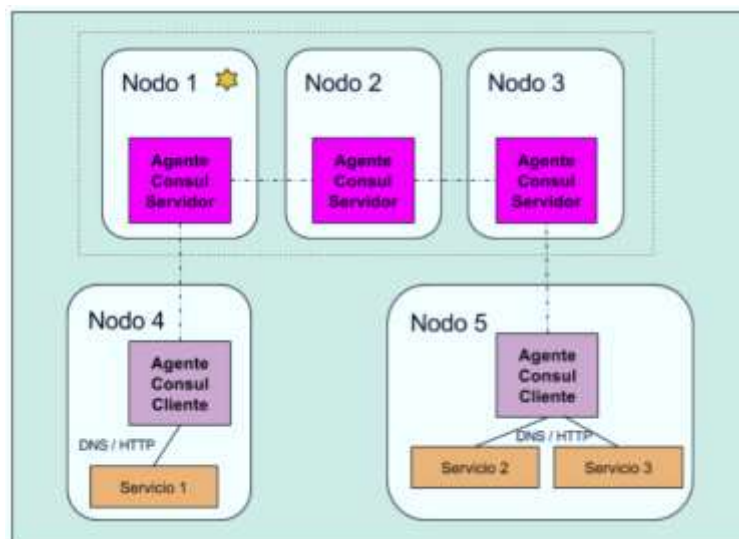
Se recomienda no tener más de 3 a 5 agentes servidores por datacenter, ya que podrían darse pérdida de datos, también es muy importante seleccionar un “líder” que es el que será el único que responderá directamente las peticiones, el resto deben enviárselo a él. Si solo hay un agente servidor, este se puede seleccionar como líder con la propiedad de Bootstrap (Paradigma Digital, 2018).

Agente Cliente. Agente muy ligero, utilizado para los nodos que deben ser inspeccionados y notificar a los agentes servidores. Se encarga de reenviar las peticiones y el resultado de actividad y del testeo de calidad el agente servidor (Paradigma Digital, 2018).

Finalmente, en la figura 13 se puede ver como generalmente está distribuido un sistema basado en Consul,

Figura 13

Sistema basado en Consul



Nota: Figura tomada de **(Paradigma Digital, 2018)**.

Es importante mencionar que ambos tipos de agentes pueden recibir peticiones. Lo ideal es tener agente cliente corriendo en el mismo nodo del servicio y que este sea encargado de las respuestas y los tests de calidad. Se lo debería hacer de esta manera para evitar que en caso de que el servicio se caiga por completo, se evite no poder acceder al resto de servicios que maneje el agente servidor, sino que solo deje de funcionar el servidor cliente auxiliar (Paradigma Digital, 2018).

Sistemas de ingeniería, informáticos y de comunicación

El objetivo principal de este trabajo de titulación es el poder manejar remotamente un sistema domótico completo. Para esto se utiliza una arquitectura de software específica, elementos o dispositivos de hardware puntuales y finalmente se los interconecta usando a las redes y protocolos de comunicación actuales. Debido a esto

surge la necesidad de definir adecuadamente las herramientas y tecnologías específicas más oportunas para lograr todos los objetivos propuestos.

Sistemas Ciber-físicos (CPS, Cyberphysical Systems)

En ingeniería, un sistema ciber-físico está construido específicamente para funcionar con la integración de algoritmos computacionales y de dispositivos electrónicos físicos. Los avances más vanguardistas en cuanto a CPS permiten escalabilidad, resiliencia, y mayor usabilidad en muchos sistemas críticos, llamados así debido a las prestaciones y a la disponibilidad ofrecen. Si se habla de CPS de manera general se puede mencionar proyectos grandes como: smart grid, carros autónomos e inclusive un sistema domótico. Pero si se habla de CPS en una escala más pequeña, puede tratarse de un sensor o un actuador configurado para funcionar con un microcontrolador específico. Un punto importante es que UN CPS no necesariamente está conectado al internet, sino pueden ser redes internas con ciertos protocolos de comunicación (NSF, 2020).

Componentes domóticos

Sensores y actuadores

Los sensores, también llamados transductores, son elementos electrónicos que se utilizan para la adquisición de datos del entorno. Los sensores por lo general convierten un fenómeno físico en voltaje, para que la información sea medible y legible con facilidad (Dewesoft, 2020). A diferencia de los sensores, los actuadores están orientados a captar algún tipo de señal o de energía externa y reaccionar dependiendo a su configuración. En pocas palabras, transforman una variable en una acción

utilizando algún mecanismo o proceso automatizado interno o perteneciente a un sistema más grande del cual formen parte.

Como se mencionó, existen diversos sensores y actuadores que son utilizados en domótica, y en la tabla 6 se definen los que se considera más oportunos para el desarrollo del proyecto.

Tabla 6

Sensores y actuadores.

Nombre del elemento	Voltaje de alimentación	Corriente consumida	Tipo del elemento
Fotocelda	100 V. (Máximo).	5mA	Sensor
Sensor de Gas MQ2	5 V.	150mA	Sensor
Zumbador pasivo	5V.	50mA	Sensor (Actuador).
PIR Motion	5 a 12 V.	60uA	Sensor
Motor DC	5 a 12V.	100mA	Actuador
Microservomotor	4.8 a 5V.	200mA	Actuador
Luces Led	5V.	30mA	Actuador
Pantalla LCD	5V	60mA	Actuador

Plataformas de Hardware

Arduino

Arduino es una compañía y un proyecto open-source, que diseña placas de desarrollo de hardware utilizadas para el desarrollo de dispositivos digitales y sistemas interactivos de muchos tipos y propósitos. Por lo general las placas Arduino usan diversos microcontroladores y microprocesadores de tipo ATMEL que van conectados bajo la configuración simple y de sistema mínimo a circuitos impresos y también pueden ser extensibles hasta shields de propósito específico que pueden ser desarrollados por empresas propias. La mayor ventaja que ofrece este tipo de plataforma, es su versatilidad y las múltiples aplicaciones electrónicas que puede brindar, además del tipo

de conexión que puede ofrecer, ya sea en redes locales por Ethernet o Wifi. Esto último claramente depende de la placa que se necesite y de las características que brinde. Tal es el caso del ESP8266 un Arduino especial que se especifica más a fondo en la tabla 7 y puede observarse en la figura 14.

Figura 14

ESP8266 – NodeMCU V.3.



Nota: Figura tomada de (Naylamp, 2021)

Tabla 7

Características del ESP8266 – NodeMCU V.3.

ESP8266 – NodeMCU V.3.	
Voltaje de Alimentación	5V
Memoria	Capacidad de SD FLASH/ 4MB
Voltaje de Salida	3.3 V
Chips integrados	<ul style="list-style-type: none"> ○ Wi-Fi Direct (P2P), soft-AP. 802.11 b/g/n. ○ Antena en PCB
Costo	Entre 8 y 10\$

Nota: Tabla tomada de (Naylamp, 2021).

Herramientas de pruebas

Cuando se desarrolla un proyecto con metodologías específicas, es muy importante realizar pruebas funcionales que involucren tanto el rendimiento como la usabilidad, esto con el fin de determinar si el sistema es capaz de cumplir con los objetivos propuestos. Hay una gran cantidad de herramientas tanto de código abierto como de pago que facilitan y brindan las características necesarias para valorar un proyecto.

Gatling

Gatling es una herramienta, diseñada como una interfaz de código abierto especializado en la realización de pruebas funcionales y de rendimiento, su enfoque es el de proporcionar soluciones eficientes y altamente calificadas sin que representen ningún costo para los desarrolladores que deseen evaluar las cargas de trabajo y las peticiones por segundo que puede soportar el sistema. La base de su funcionamiento son los scripts, que pueden adaptarse a cualquier entorno o escenario de desarrollo, además no necesitan muchas configuraciones, simplemente escalar las pruebas que se quiera realizar y especificar la cantidad de peticiones y usuarios que se quisiera. Es importante instalar un compilador para poder utilizarlo, ya sea Maven, Gradle u algún otro, también se necesita algún editor de código o cualquier IDE que permita abrir y editar los scripts específicos para cada prueba. Finalmente, la característica principal de Gatling es proporcionar una gran cantidad de información para poder realizar cualquier tipo de prueba y scripts propios, la visualización de los resultados se representa en archivos HTML generados al finalizar una prueba, en los que se detalla cada una las peticiones, usuarios y resto de variables de prueba que se haya asignado. (Gatling, 2013).

Sistema de escalas de usabilidad

El sistema de escalas de usabilidad (System Usability Scale, SUS), es una herramienta metodológica y estandarizada por múltiples instituciones y entidades tecnológicas, su propósito principal es el de medir algún objeto, proyecto, dispositivo o aplicación. La característica que lo hace sobresalir de otras metodologías de prueba o de medición, es que a pesar de su simpleza brinda una información muy específica y clara en el caso de que se quiera hacer algún cambio o actualización al proyecto.

Este sistema de medición está conformado por 10 preguntas que se pueden calificar con un puntaje del 1 al 5. El valor de 1 representa estar totalmente en desacuerdo y el valor de 5 representa estar totalmente de acuerdo, las preguntas deben adaptarse dependiendo de los requerimientos del evaluador reemplazando los espacios en blanco de cada pregunta con la palabra objeto, aplicación, sistema, dispositivo, etc. Para el proyecto lo oportuno sería escribir “sistema domótico”, haciendo referencia tanto al HMI como a las acciones que pueden ser observadas en la maqueta elaborada, pero teniendo en cuenta que puede ser replicada en un entorno real.

- Creo que usaría este [...].
- Encuentro este [...] innecesariamente complejo.
- Creo que el [...] fue fácil de usar.
- Creo que necesitaría de una persona con conocimientos técnicos para poder usar este [...].
- Las funciones de este [...] están bien integradas.
- Creo que el [...] es muy inconsistente.
- Imagino que la mayoría de la gente aprendería a usar este [...] en forma muy rápida.
- Encuentro que el [...] es muy difícil de usar.

- Me siento confiado al usar este [...].
- Necesité aprender muchas cosas antes de ser capaz de usar este [...].

La puntuación final de este sistema debe tener ciertas consideraciones, en primer lugar, no se trata de un porcentaje, sino de un puntaje sobre 100 puntos, en segundo lugar, a las preguntas impares se les debe restar 1 al puntaje, al valor de las preguntas pares debe ser restado de 5, y por último se debe sumar todos los valores para obtener el puntaje buscado. (UXpañol, 2017).

Capítulo III

Diseño de la Arquitectura

Requisitos de diseño

Como se ha mencionado en el capítulo 1 y como se observa en la figura 1, la arquitectura en su totalidad consta de tres partes principales, la primera es el Front-end con la aplicación de control, la segunda es el Backend con toda la configuración de los servicios que pueden ser orquestados y descubiertos por el agente especializado para esta tarea, además es importante mencionar que se cuenta con balanceo de cargas, es decir que habrá más de una instancia del mismo servicio, esto ayuda al sistema a gestionar los datos, ya que en caso de que una de las instancias del servicio se encuentra caída, se seleccionará la otra instancia, y no habrá inconvenientes en el envío y recepción de datos. Esta es una ventaja del descubrimiento, ya que identifica cualquier cantidad de instancias del mismo tipo y selecciona la que mejor latencia ofrece. Finalmente, la tercera parte es la casa automatizada con su microcontrolador específico siendo un sistema domótico, de este modo el trabajo cumplirá con los objetivos planteados. Además, con los estudios realizados en el capítulo 2, se pudo conocer las tecnologías, herramientas y esquemas de diseño más utilizados para este tipo de aplicación.

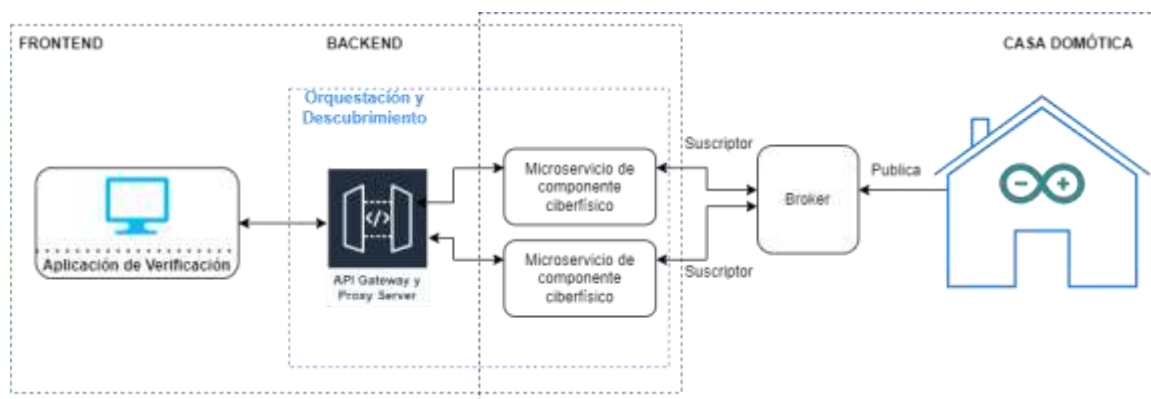
Estructura de diseño del sistema

Una vez se ha explicado de manera general la estructura de la arquitectura, se puede elaborar un diagrama donde consten todas sus partes. En la figura 15 se observa la estructura general, se tiene el Front-End con la aplicación de control, esta se comunica al Gateway del Backend, en segundo lugar se tiene el Back-end donde se albergan las configuraciones necesarias para el funcionamiento de los microservicios,

los cuales son orquestados y descubiertos por un agente especializado, y surgen a partir de componentes ciberfísicos, además el Back-end se comunica con el Broker y se gestiona los datos que se recibe del mismo, finalmente se tiene la casa domótica, donde se tiene toda la configuración de los sensores y actuadores, además de la conexión a la red y al Broker, donde se publican los respectivos mensajes en cada uno de los tópicos específicos. Es importante mencionar que el tipo de comunicación entre el Front-End y el Back-End, es distinta a la comunicación entre Back-End y la casa domótica.

Figura 15

Estructura general del diseño de la arquitectura.



Con el diagrama de la arquitectura bien definido, se puede ahondar en los requisitos tanto funcionales como no funcionales que deben tener todas las partes del proyecto en su implementación, estos se presentan en la tabla 8 y 9 respectivamente.

Tabla 8

Requisitos funcionales.

Requisito	Descripción
Requerimientos que debe cumplir la arquitectura de software. (Back-End).	<ul style="list-style-type: none"> Cada uno de los microservicios deben ser independientes, ya que representan a los componentes

Requisito	Descripción
Funcionalidades que debe cumplir la arquitectura de software. (Back-End).	<p>ciberfísicos que serán orquestados y descubiertos.</p> <ul style="list-style-type: none"> • Se debe tener una base de datos independiente para cada componente ciberfísicos, y estas pueden ser accedidas sin restricción. • Las bases de datos deben permitir almacenar cualquier cantidad de datos. • Se debe disponer un Gateway-Proxy para la comunicación tanto externa como interna de los microservicios. • Lo más importante es disponer de un agente capaz de orquestar y descubrir nuevos componentes. • Debe poder comunicarse con el Front-End y además poder obtener los datos que la casa domótica envía al bróker.
Detalles técnicos y manejo de datos.	<ul style="list-style-type: none"> • La comunicación del front-end con el back-end e RESTful ya que maneja el protocolo HTTP. • La comunicación de la casa domótica con el back-end es con el protocolo MQTT. • Todos los datos que serán de tipo JSON. • Todos los elementos de la arquitectura tanto de front-end deben contar con una conexión a una red para que cumplan con su funcionamiento.

Tabla 9*Requisitos no funcionales.*

Requisito	Descripción
Cualidades que la aplicación de verificación puede tener. (Front-End).	<ul style="list-style-type: none"> • La interfaz debe dividirse de tal manera que resulte sencillo la visualización del estado actual y el historial de los actuadores de la casa domótica. • El tiempo de respuesta de las peticiones debe cumplir con una buena QoS, esto se configura tanto en el Broker como en el Backend. • La aplicación desplegara alertas que, dependiendo de la información, además también puede desplegar un historial.
Cualidades que la arquitectura de software puede tener. (Back-End).	<ul style="list-style-type: none"> • Las bases de datos pueden estar en el entorno local o en un entorno remoto. • Cada microservicio puede estar albergado en el entorno local o en un entorno remoto. • Los microservicios pueden funcionar independientemente y se comunican con el Gateway-Proxy.
Restricciones generales.	<ul style="list-style-type: none"> • El acceso a las bases de datos es directamente desde la aplicación de control. • Ya que es una aplicación web, se debe contar con un navegador web. • Los componentes deben estar conectados a la misma red. • La QoS depende de la velocidad de respuesta de los componentes.

Requisito	Descripción
	<ul style="list-style-type: none">• La autonomía depende de la fuente de alimentación que se utilice.

Frontend

El Frontend estará representado en su totalidad por una aplicación de verificación que funcionara en cualquier navegador web y donde se puedan verificar los datos que se tenga guardados en las bases de datos específicas de cada servicio en el backend, esto ayudará a que se pueda probar la orquestación y descubrimiento que se debe implementar para los elementos del backend, además del balanceo de carga ya que se podrá evidenciar si esta característica está funcionando cuando la instancia de un servicio falle, la aplicación cuenta de tres partes: estructura, diseño y funcionalidad. Las características de la aplicación están definidas por la cantidad de componentes ciberfísicos que se tenga, ya que, cada uno de los elementos que se incluyan en la aplicación de verificación, tienen relación directa con los componentes ciberfísicos y son únicos, de este modo en la implementación se definirá la cantidad específica de elementos con los que se debe contar.

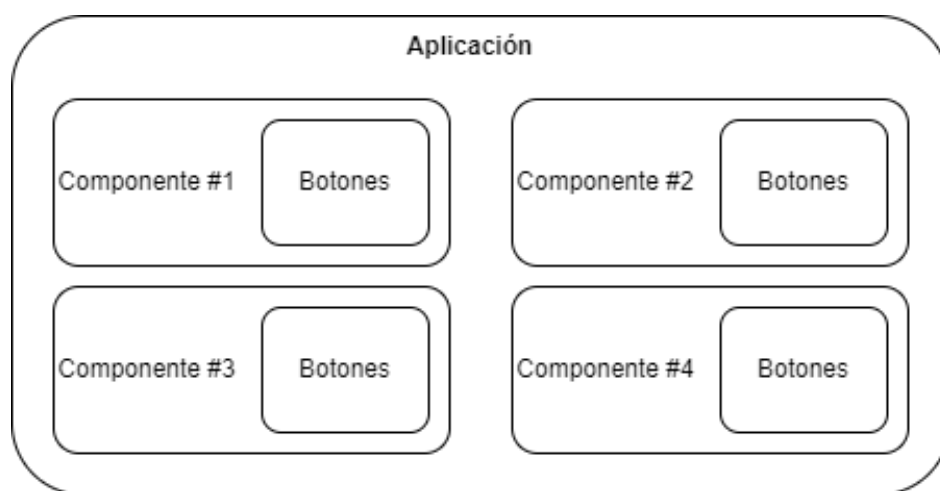
Estructura de la aplicación de verificación

La aplicación de verificación puede funcionar en cualquier navegador web y la distribución de sus elementos debe permitir al usuario interactuar con ellos de manera fluida para poder cumplir con cualquier requerimiento, cada uno de los elementos contará con botones, los que están relacionados con los componentes ciberfísicos de la casa domótica, como con los servicios del backend. En la figura 16 se presenta un bosquejo de la aplicación, la cual tendrá los componentes visuales y los botones

necesarios para cumplir con los objetivos propuestos, en este caso se tiene 4 componentes asumiendo que solo existiesen 4 componentes ciberfísicos, pero esto puede variar una vez se implementen todas las partes del proyecto.

Figura 16

Bosquejo de la aplicación de verificación.



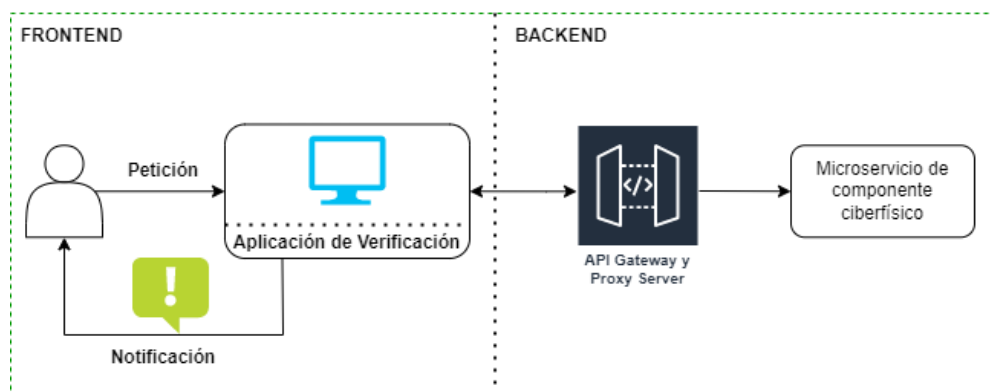
Funcionamiento de los componentes de la aplicación de verificación.

Todos los elementos de la aplicación deben funcionar de manera similar, y su propósito es obtener los datos respectivos a cada componente ciber físico de la casa domótica. Cada componente de la aplicación de control se comunicará con el Back-End, de donde se consultarán los datos necesarios, que se encuentran guardados en la base de datos de cada uno de los servicios. En la figura 17 se puede ver un diagrama de la comunicación entre Frontend y Backend, En primer lugar, se pasará por una puerta de enlace o Gateway y de ahí todas las peticiones pasaran al servicio correspondiente.

Controles de los componentes de la aplicación. Cada control o botón que se incluya en un componente específico de la aplicación, será el encargado de conectarse al Backend por las peticiones que generará utilizando su configuración de funcionalidad, de este modo se podrá cumplir con la tarea específica que se requiera, por ejemplo, conocer cuáles han sido los últimos estados del componente ciberfísicos o cual es el estado actual. Los botones estarán en la parte derecha de cada componente y contarán

Figura 17

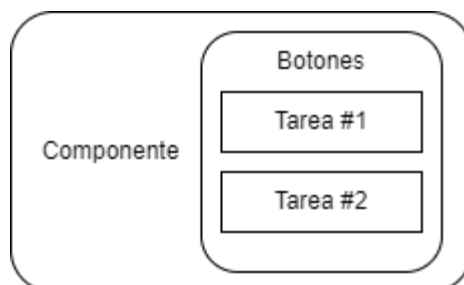
Funcionamiento de los componentes de la aplicación de verificación.



con una descripción que está relacionada con la tarea específica que estos puedan cumplir. En la figura 18 se puede ver un bosquejo de un componente con sus controles específicos.

Figura 18

Ejemplo de controles de un componente.



Componente de las alertas de la aplicación de verificación. Si bien las alertas no podrían visualizarse directamente, son un elemento muy importante, ya que indican la información que se consultó. Las alertas serán emergentes, es decir cuando se presione un botón específico de algún componente, esta se desplegará e indicará la información de tal manera que se pueda solventar todas las dudas de la petición requerida. En la figura 19 se puede ver un ejemplo de cómo debería verse una alerta emergente.

Figura 19

Ejemplo de alerta emergente.



Backend

Si bien la aplicación de verificación se podría reemplazar por cualquier tipo de servicio capaz de enviar peticiones, el Backend por otro lado no puede sustituirse tan fácilmente, ya que es el núcleo de toda la arquitectura, y es la parte fundamental del presente proyecto ya que se encargará tanto de la orquestación y descubrimiento de los componentes ciberfísicos de la casita domótica como también de la comunicación con la misma como también con el Frontend. En el Backend se tiene todos los microservicios importantes, así como las librerías externas encargadas de las

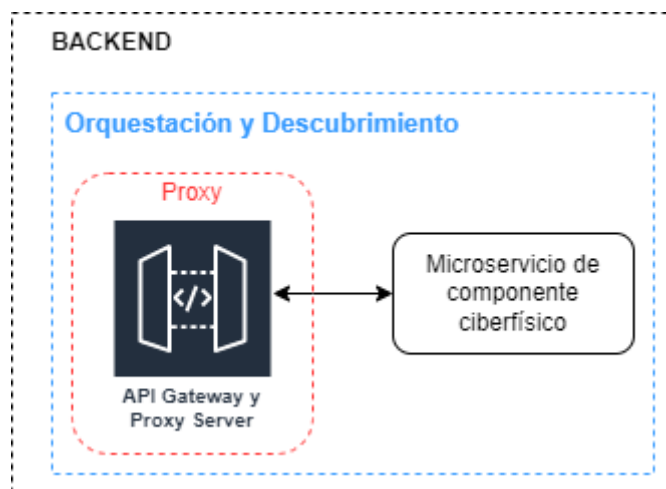
configuraciones importantes. Además del motor encargado de descubrir los componentes ciberfísicos, así como de orquestarlos.

Estructura general del backend

En el Backend se encuentran varios elementos que son muy importantes, lo que se busca es la orquestación y descubrimiento de componentes ciberfísicos, y esto se logra por las configuraciones que se hacen en el mismo. En la figura 20 se pone a vista la estructura general del backend, inicialmente cada uno de los microservicios tienen que contar con las configuraciones necesarias como: las que permiten puedan ser identificados por el agente que se encarga de orquestarlos y descubrirlos, es importante mencionar que todos los microservicios deben funcionar a través de este, para que no existan incompatibilidades de comunicación, de este modo se tiene al API Gateway que también función como Proxy y al resto de microservicios relacionados con los componentes ciberfísicos.

Figura 20

Estructura general del backend

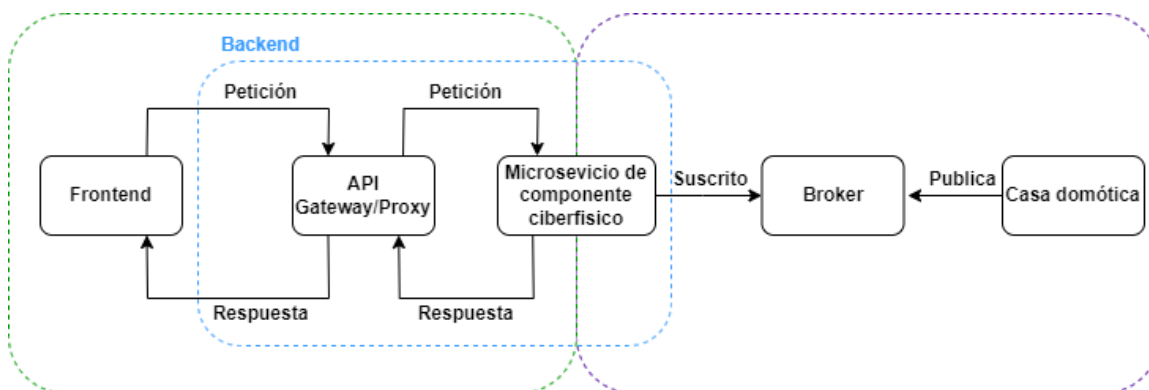


Funcionamiento general del backend

En la figura 21 se puede ver un diagrama de su funcionamiento, su comunicación con el Frontend consiste en recibir las peticiones que se hagan, estas llegan al Gateway, donde se enrutan al microservicio relacionado al componente ciber físico respectivo, donde este responde con la información que se requiera, después se tiene la comunicación con la casa domótica, donde esta se la realiza mediante un broker de mensajería que es donde se reciben las cadenas de datos y van pasando dependiendo de su orden de llegada y se almacenan en la base de datos del microservicio del componente ciber físico respectivo que este suscrito al tópico específico donde se publicó algún dato por parte del controlador de la casa domótica.

Figura 21

Diagrama de flujo del funcionamiento general del backend



Estructura interna de los microservicios

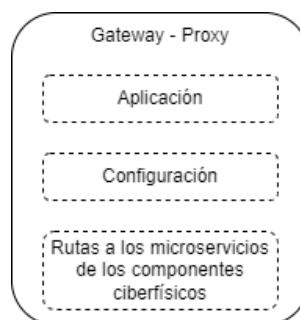
Los microservicios cumplen un rol un fundamental para que todo el sistema funcione de tal manera que se cumpla con el propósito planteado. Se tiene el servicio que cumple con el papel de Gateway y Proxy, también se tiene los microservicios correspondientes a cada uno de los componentes ciberfísicos, que son los que se suscriben al bróker y finalmente se tiene al agente encargado de orquestar y descubrir,

que, si bien no es un microservicio como tal, es el motor y la base del resto de microservicios, ya que de él depende su funcionamiento y además saber el estado de los mismos.

Servicio de Gateway y Proxy. Este microservicio, como su nombre lo indica, es la puerta de enlace y el intermediario del cliente con el servidor, en este caso el cliente es el usuario que envía las peticiones usando al frontend y estas llegan a la puerta de enlace donde se enrutan al servidor, que puede ser cualquiera de los microservicios de los componentes ciberfísicos, dependiendo de la petición que se haga. El Gateway al comunicarse únicamente con el frontend, funciona únicamente bajo un protocolo de comunicación específico. En la figura 22 se puede ver la estructura de este microservicio.

Figura 22

Estructura del servicio de Gateway y Proxy.



Microservicios de los componentes ciberfísicos. Una vez la petición pasa a través del Gateway, estos microservicios son los encargados de recibirla y gestionar la orden requerida, además son los encargados de establecer la conexión con el Broker, a través de una suscripción a los tópicos específicos del mismo. Estos microservicios están relacionados con los actuadores de la casa domótica.

Estructura de los microservicios de los componentes ciberfísicos. Su

estructura puede verse en la figura 23, un microservicio debe tener varios paquetes y clases. En primer lugar, se tiene el paquete de aplicación, con sus clases de configuración, que pueden ser varias. En segundo lugar, se tiene el paquete de la entidad con su clase propia, en tercer lugar, se tiene el controlador, donde se encuentran definidas las rutas a donde apuntan las peticiones realizadas desde el frontend, como cuarto punto, se tiene el objeto de acceso a datos, donde se define el repositorio dentro del microservicio y habilita el uso de sus funciones, en quinto lugar, se tiene el paquete del servicio como tal, donde está su interfaz con su respectiva implementación. Finalmente se tiene un archivo especial llamado propiedades, donde se encuentra toda la configuración para que pueda ser orquestado y descubierto el servicio, toda la información acerca de la base de datos y todos los atributos importantes como el nombre del servicio y el puerto de la red en el que funcionará.

Funcionamiento de los microservicios de los componentes ciberfísicos.

En cuanto, al funcionamiento de un microservicio de este tipo, el controlador y la implementación del servicio como tal están relacionados al envío y la recepción de datos, el paquete de la entidad le brinda las características de las variables a la clase y a los elementos de datos que se manejarán en el respectivo microservicio. La parte de aplicación configuración sirven para que se identifique como una aplicación y también para que pueda conectarse al Broker. El proceso interno comienza con la petición que envía el servidor proxy, a continuación, el controlador junto al resto de sus características heredadas o adquiridas del resto de clases, recibe la información, gestiona los datos y los envía, y en el caso de configuración, se obtiene los datos del

Broker y esto se guarda en la base de datos respectiva. En la figura 24 se puede ver un diagrama del funcionamiento interno de un microservicio.

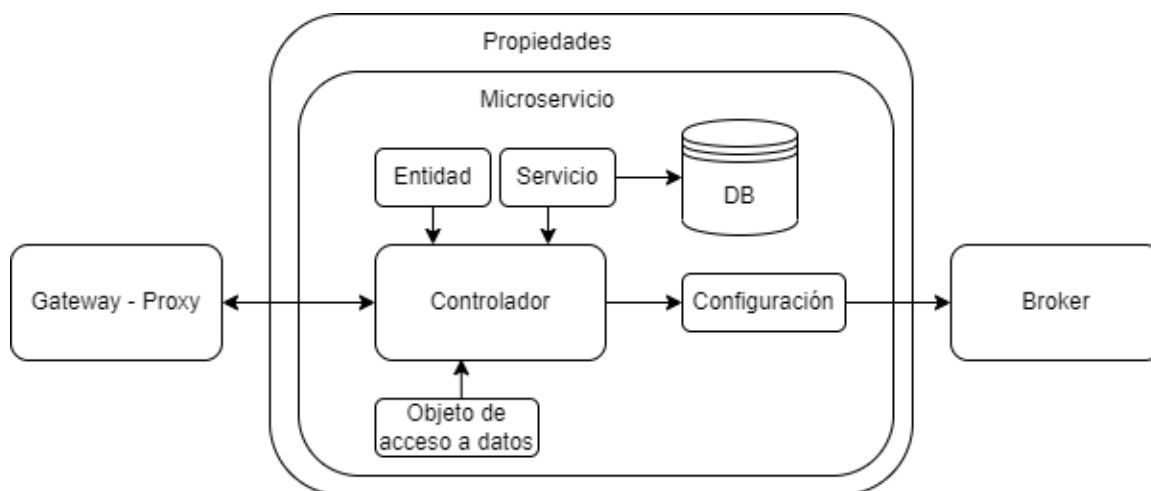
Figura 23

Estructura de un microservicio.



Figura 24

Funcionamiento de un microservicio.



Agente orquestador y de descubrimiento.

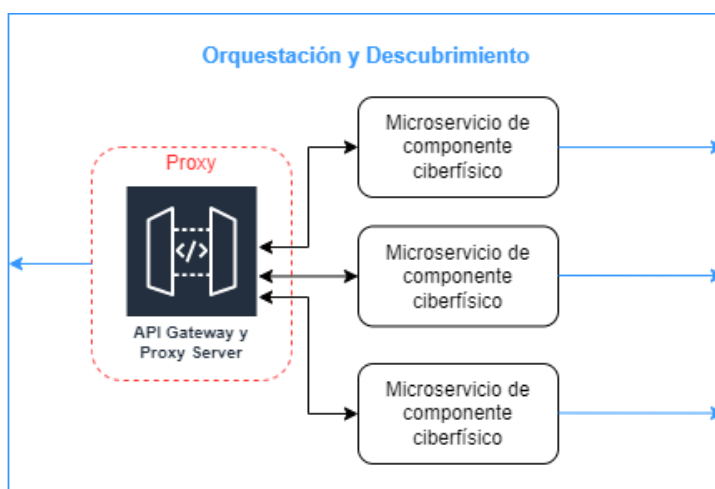
Para poder demostrar la aplicabilidad práctica de todo el sistema y sobre todo del enfoque que se le quiere dar al trabajo, lo más importante es mencionar todo acerca del agente orquestador y encargado del descubrimiento de los componentes ciberfísicos, esto permitirá tener un sensado continuo del estado de cada uno de los microservicios, además de una fácil gestión en el caso de que exista algún problema con alguno de estos componentes. Además, una de las ventajas más importantes es que este agente puede ser levantado localmente o remotamente.

Estructura del agente orquestador y de descubrimiento.

La estructura del agente como tal y su configuración es algo que funciona en segundo plano, la importancia de su estructura radica cuando se lo utiliza para que orqueste y descubra servicios existentes, en la figura 25 se puede ver este comportamiento, donde todos los microservicios sin excepción están registrados y conectados al agente.

Figura 25

Estructura del agente orquestador y de descubrimiento.

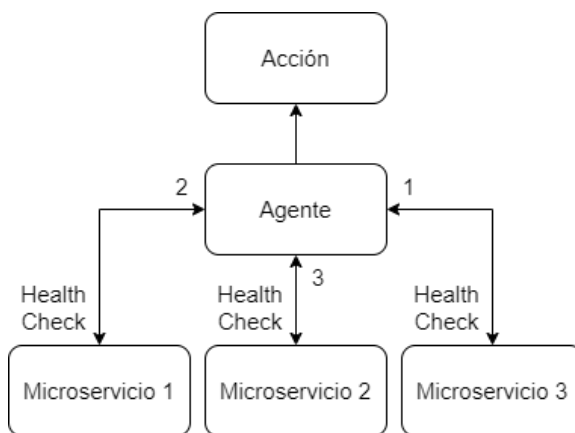


Funcionamiento del agente orquestador y de descubrimiento.

El funcionamiento del agente consiste en identificar los microservicios, en este caso los que están relacionados con los componentes ciberfísicos, una vez identificados, su siguiente función consiste en orquestar las acciones, es decir va indicando cual es el microservicio que debe realizar la petición sin importar ningún orden. Además, el agente tiene una interfaz propia donde se puede verificar el estado de cada microservicio, con una función llamada “Health Check” que es similar a ir enviando pings cada cierto tiempo. En la figura 26 se ilustra el funcionamiento del agente.

Figura 26

Funcionamiento del agente orquestador y de descubrimiento.



Casa domótica

Para poder demostrar el funcionamiento de toda la arquitectura de manera práctica es importante contar con un medio físico, con el cual se pueda obtener datos con algunos sensores, para que algunos actuadores puedan cumplir con cierta tarea predeterminada. Además de esto es importante mencionar que las acciones que los

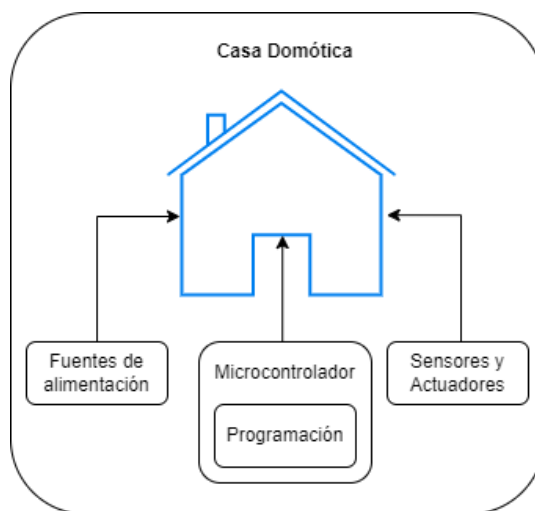
actuadores realicen serán enviadas al Backend, publicando la información en tópicos específicos de un Broker para cada componente ciber físico.

Estructura interna de la casa domótica

Internamente la casa domótica debe contar con un microcontrolador, sensores, actuadores y las fuentes de alimentación necesarias. En la figura 27 se puede ver la estructura, además internamente en el microcontrolador deben estar programados todos los componentes, con sus respectivas acciones y con la configuración necesaria para que se puedan enviar y publicar los datos en tópicos específicos del Broker.

Figura 27

Estructura de la casa domótica.



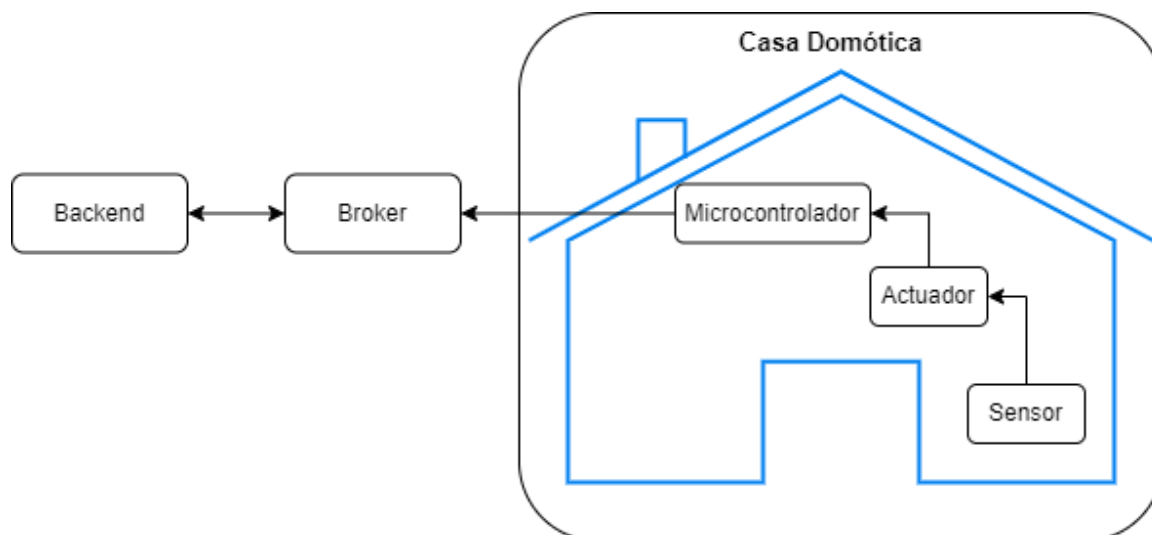
Funcionamiento de la casa domótica

Como ya se ha mencionado, el funcionamiento de la casa se basa en la programación que se realice en su microcontrolador, en este caso se debe tener la configuración tanto de los sensores como de los actuadores y lo más importante las

configuraciones necesarias para que se logre conectar tanto a una red local y también poder conectarse al Broker, para poder publicar la información de cada uno de los actuadores en los tópicos correspondientes para que después los microservicios del Backend suscritos a dichos tópicos, puedan recibir la información y guardarla en la base de datos respectiva. La generación de la publicación hacia el broker, se basa en las acciones que realicen los sensores y actuadores, una acción de un sensor, activará o desactivará un actuador, entonces de este modo, a partir de la lógica del microcontrolador, se generaran los datos necesarios para ser enviados al broker y posteriormente, estos datos sean recibidos por el backend. En la figura 28 se puede ver el funcionamiento de manera más gráfica.

Figura 28

Funcionamiento de la casa domótica.



Capítulo IV

Implementación de la Arquitectura

Después de haber definido todas las consideraciones de diseño y funcionamiento básico que deben tener cada uno de los componentes de la arquitectura, y además debido a la investigación y la información previamente recopilada en el capítulo 2, ya se pueden establecer las herramientas y tecnologías más adecuadas para poder cumplir con los requerimientos y objetivos del proyecto.

Debido a los diagramas que se ha ido utilizando, se puede seguir desarrollándolos para que se pueda visualizar el proceso y la evolución de los mismos, a fin de tener una perspectiva completa y final de todo el sistema, y sobre todo poder entender su funcionamiento total con cada uno de los componentes desplegados en la red y funcionando correctamente.

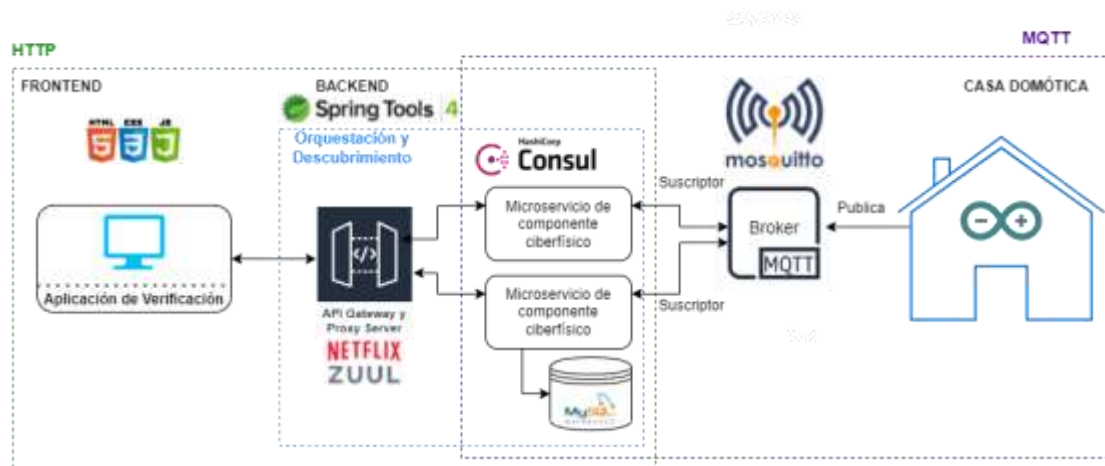
Estructura general de la arquitectura implementada

Lo más importante es conocer que herramientas y tecnologías son las que permiten el funcionamiento del sistema, y que son las más adecuadas para implementar la arquitectura en su totalidad. En la figura 29 se observa toda la estructura implementada, en primer lugar, se tiene la aplicación de verificación en el Frontend, para su desarrollo se utiliza HTML, CSS y JavaScript, siendo así *responsive*, y además muy bien organizada en distribución, diseño y funcionamiento. En la parte del Backend, para el desarrollo de los microservicios de los componentes ciberfísicos se utiliza STS4, Spring, que funciona bajo el lenguaje de programación Java, para el Gateway y Proxy de utiliza Netflix Zuul, para la gestión y control de las bases de datos, se utiliza para MySQL Workbench y lo más importante, para la parte de orquestación y descubrimiento

de los componentes, se utiliza Hashicorp Consul, que es un motor especializado en el desarrollo de estas tareas, ya que su configuración es simple y su interfaz es de fácil utilización y muy intuitiva. Para la parte del Broker, se utiliza Mosquitto de Eclipse, ya que, es Open Source y su configuración y utilización es bastante sencilla. Por último, se tiene a la casa domótica, en ella se utiliza como microcontrolador, un módulo ESP8266-NodeMcu V3, basado en Arduino y con la gran ventaja de que tiene un módulo WiFi, que permite la conexión a internet, además se tiene la fuente de alimentación y los sensores y actuadores necesarios para cumplir con los objetivos planteados.

Figura 29

Estructura general de la arquitectura implementada



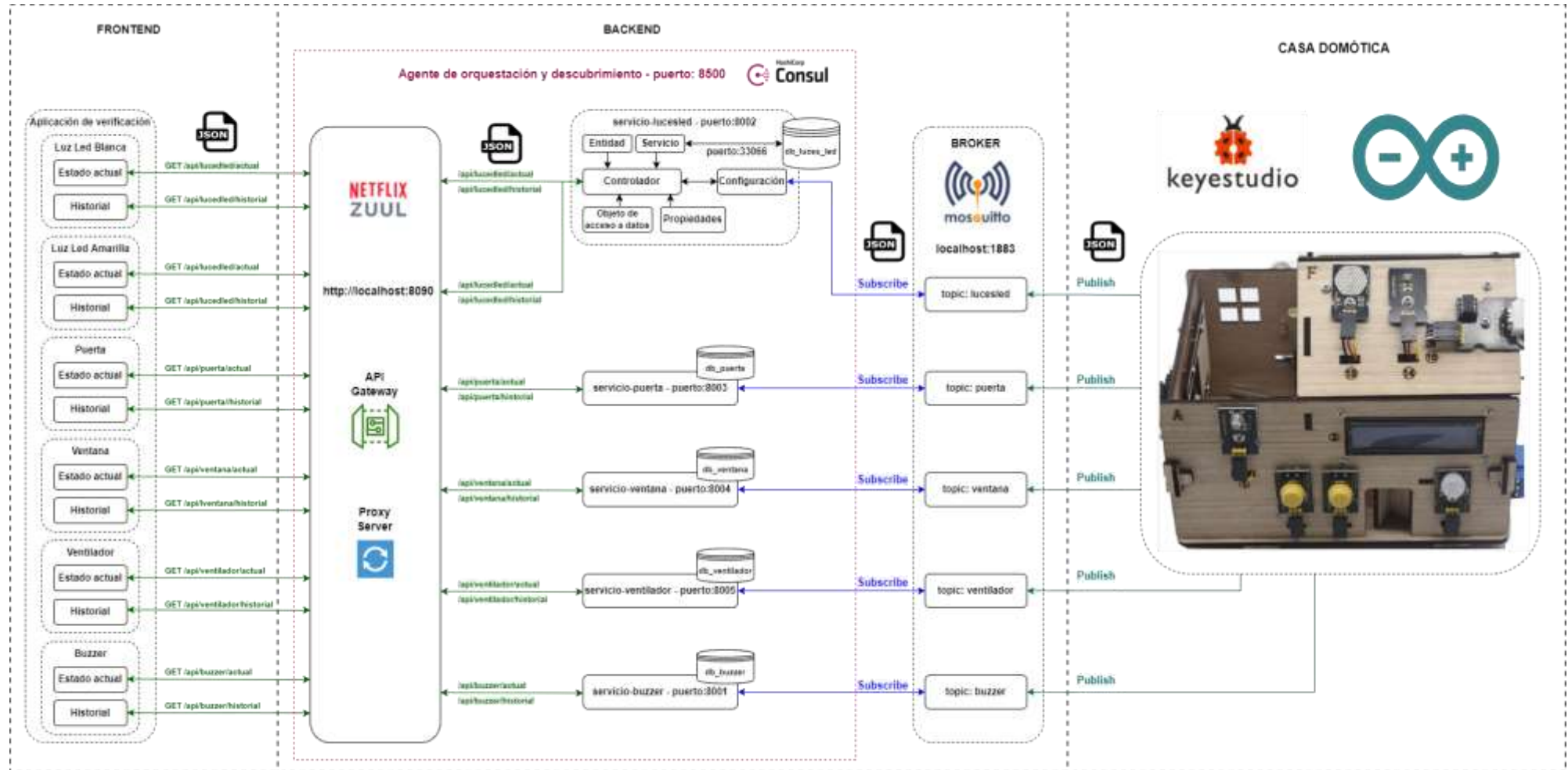
Arquitectura final.

Es importante definir cuál es la arquitectura final, con el fin de entender su funcionamiento completo antes de ir abordando a detalle cada uno de sus elementos, en la figura 30 se puede apreciar la arquitectura final, donde en primer lugar se tiene el Frontend, la aplicación de verificación que puede funcionar en cualquier navegador web y desarrollada con HTML, CSS y JavaScript, esta consta de seis componentes: luz led blanca, luz led amarilla, puerta, ventana, ventilador y buzzer, dentro de cada uno de

estos componentes se tiene dos botones, que tienen una pequeña descripción respecto a la tarea que realizan, el primero es para ver el estado actual del componente ciber físico al que están relacionados y el segundo es para ver el historial de acciones de dicho componente, cada botón está configurado en su parte funcional para que genere y envíe las peticiones HTTP de tipo GET al Gateway y este dependiendo del endpoint específico, envía la petición al microservicio del componente ciber físico que corresponda, y entonces se devuelve la información requerida al usuario donde podrá visualizar una notificación dependiendo de los datos que haya obtenido, en segundo lugar se tiene al Backend, una arquitectura de microservicios desarrollada con Spring Tools Suite 4, el primer microservicio importante es el que cumple el papel de Gateway y Proxy, este se implementa utilizando la librería open Source de Netflix llamada Zuul, por defecto funciona en el puerto 8090, el resto de microservicios son los relacionados a los componentes ciberfísicos: servicio-lucesled, servicio-puerta, servicio-ventana, etc. Cada uno funciona en el puerto específico que se defina, internamente tienen los paquetes que le dan su funcionalidad: entidad, servicio, controlador, dao, propiedades y configuración, también cada uno dispone de su propia base de datos, las cuales funcionan a través del puerto 33066, todos los microservicios funcionan con el agente de descubrimiento y orquestación Consul que esta en el puerto 8500, y que es la base del funcionamiento del Backend, ya que, en caso de no encontrarse habilitado, todos los microservicios que necesiten ser descubiertos y orquestados, no funcionarán, además cada uno de estos microservicios tienen el rol de estar suscritos a tópicos de un bróker de mensajería que funciona en el puerto 1883, donde la casa domótica que funciona con un microcontrolador Arduino publica la información dependiendo de las acciones de los componentes ciberfísicos y una vez los datos son recibidos en el Backend, estos se guardan en bases de datos específicas de cada uno de los microservicios.

Figura 30

Arquitectura final implementada.



Descripción de la casa domótica

La casa utilizada para la implementación de la domótica en la misma es de la marca Key-Studio, una empresa que fabrica y distribuye este tipo de Kits para el área recreativa y académica, en especial este Kit está orientado a funcionar con Arduino para ser el controlador de la casa, sin embargo, el controlador y el shield propios de la compañía serán utilizados únicamente para alimentar los sensores y actuadores. Para dar órdenes como tal, se utiliza un módulo ESP8266, con el que también se tiene la facilidad de conectarse a una red mediante WiFi. En la figura 31 se puede observar cómo luce la casa físicamente.

Figura 31

Casa domótica de Key-Studio.



Estructura externa de la casa domótica

Dentro del Kit para la implementación de la casa, se sitúan tanto los actuadores y sensores necesarios para su correcto funcionamiento, tanto cables como elementos mecánicos como tornillos y tuercas, para que todo pueda juntarse adecuadamente. De

este modo se abordará la casa de tal manera que se aborden todos los elementos utilizados.

Sensores. Los sensores utilizados son los siguientes: pulsadores, fotocelda, sensor de gas MQ2 y un sensor de movimiento tipo PIR. En la figura 32 se puede apreciar cada uno de estos componentes, importantes para que puedan funcionar correctamente todas las acciones necesarias de los actuadores y él envié de datos al Broker.

Actuadores. Sin duda otros componentes muy importantes son los actuadores, esto ya que son los encargados de realizar alguna acción en caso de que uno de los sensores se active o desactive. Los actuadores utilizados son los siguientes: Luces Led, Ventilador, LCD, Servomotores de puerta y ventana, y un buzzer. En la figura 33 se puede observar los actuadores, a excepción de los que se encuentran internamente los cuales son el servomotor de la puerta y el buzzer.

Figura 32

Sensores de la casa domótica.

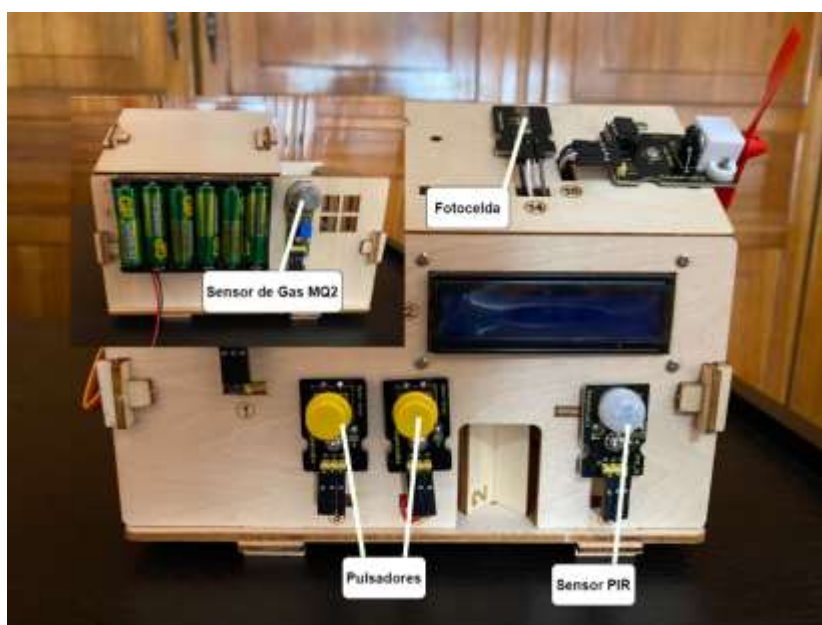
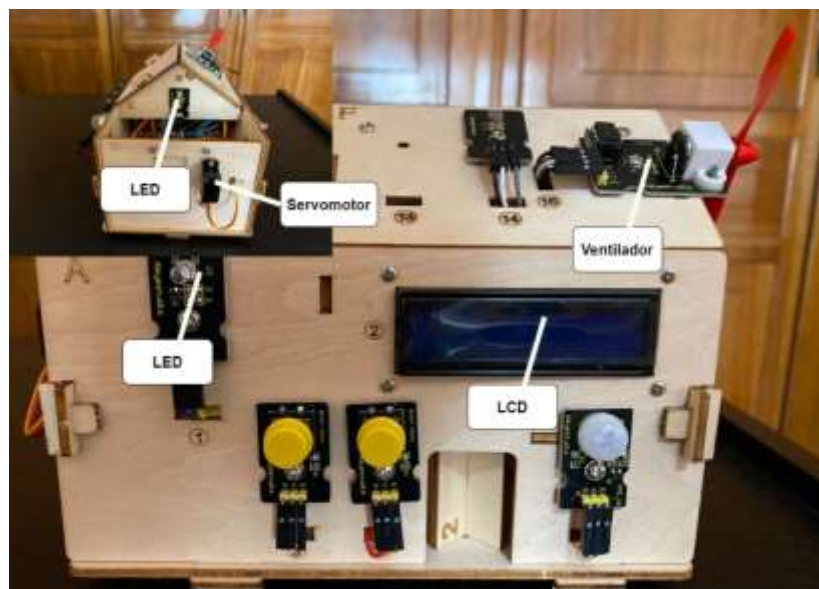


Figura 33

Actuadores de la casa domótica.



Estructura interna de la casa domótica

La casa domótica internamente consiste de las conexiones de los sensores y actuadores al controlador y al shield que será la fuente de alimentación de los mismos. El propósito de este capítulo es profundizar en los detalles importantes de funcionamiento, más no en temas como el ensamblaje, en la figura 34 se puede ver un diagrama de cómo se encuentran las conexiones. En primer lugar, se tiene al Arduino propio de la empresa y su Shield, como núcleo importante para la alimentación de los elementos, después se tiene el módulo ESP8266, que es el controlador encargado de gestionar la toma de datos de los sensores y las acciones importantes de los actuadores. Todos los sensores a excepción de la fotocelda son digitales, esto es algo que se debe tener en cuenta, ya que el módulo ESP solo dispone de una entrada analógica. Además, solo tiene 10 pines que pueden ser utilizados como entradas y salidas, por lo que limita la cantidad de componentes que se puede utilizar.

Figura 34

Estructura interna de la casa domótica.



Especificaciones técnicas de los componentes de la casa domótica. Un objetivo importante es conocer cada uno de los elementos que se utilizan, la cantidad necesaria y el rol específico que cumplen. De este modo se puede tener una idea de la energía consumida y las prestaciones disponibles. En la tabla 21 se presenta un listado resumido de los componentes electrónicos utilizados en la casa domótica con algunas de sus especificaciones técnicas más importantes. Es importante mencionar que todos los elementos vienen dentro del propio Kit de ensamblaje y están limitados a las corrientes y voltajes que el propio Shield es capaz de proporcionar.

Tabla 10

Especificaciones técnicas de los componentes de la casa domótica.

Componente electrónico o módulo	Voltaje de alimentación	Corriente consumida	Característica extra
ESP8266 - NodeMCU V.3.	5 V.	2 A.	Cantidad: 1.
Arduino UNO.	5 V.	2 A.	Cantidad: 1.
Sensor Shield	5 V.	2 A.	Cantidad: 1.
Micro-servomotores.	4 – 6 V.	150 – 250 mA.	Cantidad: 2. Rotación: 180°.
Módulo de sensor de gas MQ2.	2 – 5 V.	2 – 5 mA.	Cantidad: 1.
Pulsadores	5 V.	3 – 5 mA.	Cantidad: 2.

Luz Led	4.5 V – 5V.	20 mA.	Cantidad: 2. Color Blanco y Amarillo.
Fotocelda	2 – 5 V.	2 – 5 mA.	Cantidad: 1.
Módulo de sensor de movimiento PIR.	2 – 5 V.	25 mA.	Cantidad: 1.
Buzzer	2 – 5 V.	10 mA.	Cantidad: 1.
Ventilador	2 – 5 V.	50 mA.	Cantidad: 1.
LCD	2 – 5 V.	100 mA.	Cantidad: 1.
Pila AAA	Voltaje que suministra: 1.5 V.	Corriente que suministra: Hasta 900 mAh.	Cantidad: 6. Requiere portapilas en serie para las 6.

Por añadidura, se puede hacer referencia a que el Shield contiene un Switch con el que se puede apagar o encender el sistema de la casa domótica.

Conexiones de los componentes de la casa domótica. Una vez bien definidas las especificaciones técnicas de todos los componentes utilizados, las conexiones pueden observarse de mejor manera, en primer lugar, en la figura 35 se puede ver el Arduino UNO a la izquierda y el Shield a la derecha, y que son utilizados para alimentar a los sensores y actuadores. Estos dos elementos se conectan juntos, el Shield sobre el Arduino. El Shield se encarga de gestionar las corrientes y voltajes, para que cada componente funcione correctamente. Para la parte de control se utiliza el módulo ESP8266, y a este módulo se conectan únicamente los pines de señal de control de cada uno de los componentes. Finalmente, las fuentes de alimentación, tanto del Arduino UNO como del ESP8266 pueden ser diferentes, en ese caso es importante que la tierra sea común, para que el circuito se cierre y todo funcione correctamente, sin ningún problema con los voltajes, sobre todo del controlador que necesita conocer específicamente el nivel de voltaje digital, ya sea HIGH o LOW.

Una vez conocidas todas las conexiones que tienen los sensores y actuadores con el Shield y el módulo ESP8266, para fines prácticos, es importante conocer a que pines de control están conectados, ya que esto facilita la comprensión de las conexiones y la utilización de los pines dentro del código, además de saber si cada uno de los pines representa una entrada INPUT, una salida OUTPUT, asimismo el tipo de comunicación, puede ser PWM o HIGH - LOW haciendo alusión a los voltajes altos y bajos. En la tabla 11 se puede ver los pines, el tipo de comunicación que ocupan cada uno de los sensores y actuadores, si son definidos como entradas o salidas, y sobre todo el GPIO que representan dentro del código. Conocidos todos los pines a los que están conectados cada módulo, en la figura 37 se puede observar el esquemático del circuito con todos los módulos y las conexiones a los respectivos pines, tanto de alimentación en el Shield del Arduino UNO como de control en el ESP8266.

Figura 36

Conexiones de los componentes electrónicos.

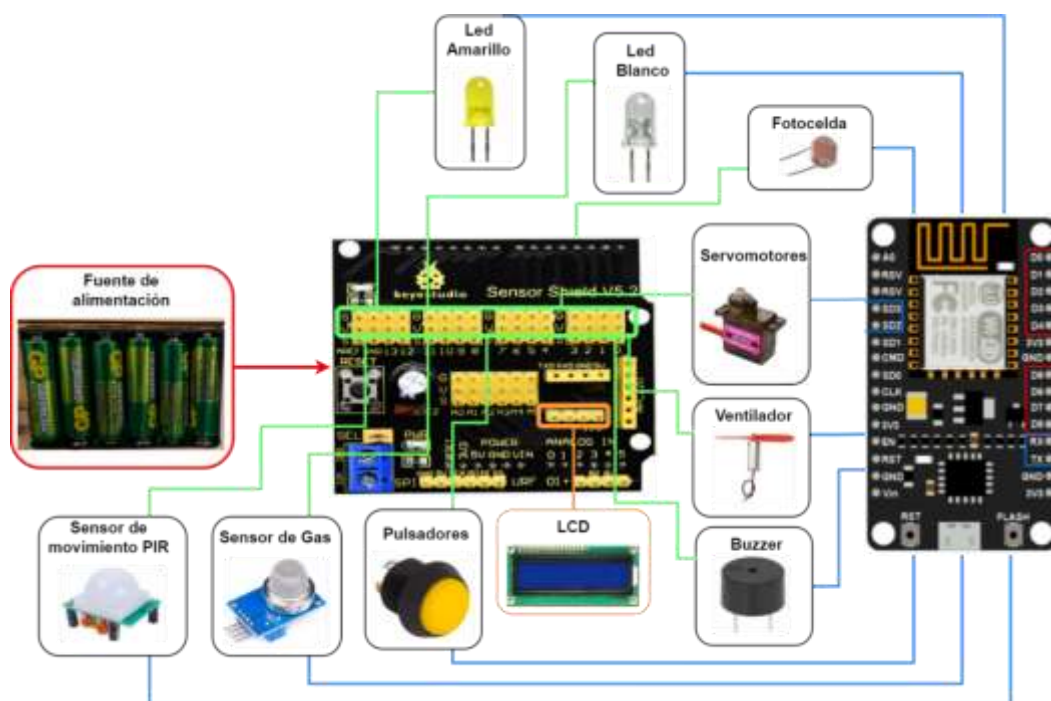


Tabla 11

Componentes electrónicos con sus pines y su tipo de comunicación.

Componente electrónico	Pines	GPIO	Tipo de comunicación
Led Blanco	D0	16 - OUTPUT	HIGH - LOW
Led Amarillo	D4	2 - OUTPUT	HIGH - LOW
Pulsado Encendido	D6	12 - INPUT	HIGH - LOW
Pulsado Apagado	D9	3 - INPUT	HIGH - LOW
Servomotor Puerta	TX	1 - OUTPUT	PWM
Sensor PIR	D7	13 - INPUT	HIGH - LOW
Servomotor Ventana	D3	0 - OUTPUT	PWM
Sensor Gas	D5	14 - INPUT	HIGH - LOW
Ventilador	D1	4 - OUTPUT	HIGH - LOW
	D2	5 - OUTPUT	PWM
Buzzer	D8	15 - OUTPUT	HIGH - LOW
Fotocelda	A0	ANALOG INPUT	Analógica
LCD	Pines I2C del Arduino UNO	OUTPUTS	I2C

Funcionamiento de la casa domótica

Una vez que se ha definido todas las partes que tiene la casa domótica, hay que saber que rol cumplen cada una de las mismas y saber que pueden ofrecer. Entonces en la tabla 12 se tiene una tabla con la acción que realiza la casa domótica, las partes encargadas y una pequeña descripción de cómo es que se da esta acción.

Figura 37

Esquemático del circuito.

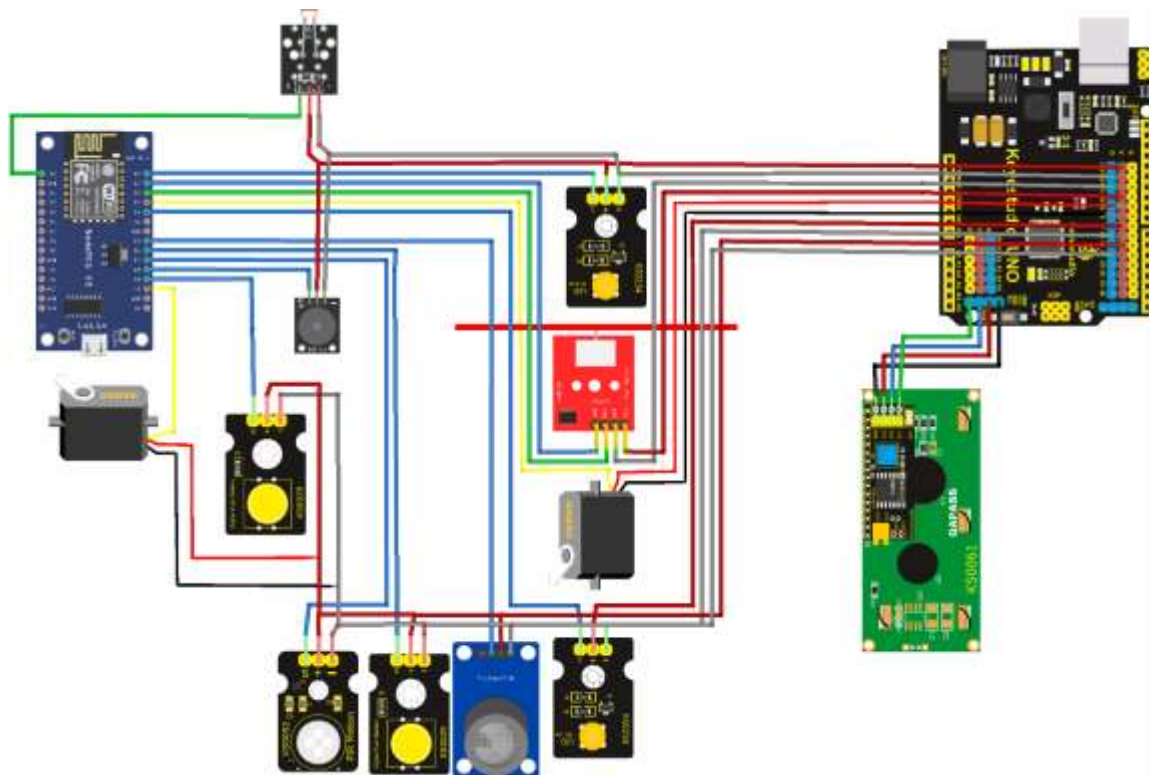


Tabla 12

Funcionamiento de la casa domótica.

Acción de la casa domótica	Parte o partes encargadas de la acción	Descripción
Encender Luces Led	<ul style="list-style-type: none"> • Pulsador de encendido. • Led Blanco. • Led Amarillo. 	Las luces led se encienden cuando se presiona el pulsador de encendido.
Apagar Luces Led	<ul style="list-style-type: none"> • Pulsador de apagado. • Led Blanco. • Led Amarillo. 	Las luces led se apagan cuando se presiona el pulsador de apagado.
Abrir y Cerrar Ventana	<ul style="list-style-type: none"> • Servomotor de ventana. • Sensor de Gas MQ2 	La ventana se abre cuando el sensor identifica algún tipo de Gas. La ventana se cierra cuando el sensor deja de identificas el Gas.

Acción de la casa domótica	Parte o partes encargadas de la acción	Descripción
Abrir y Cerrar Puerta	<ul style="list-style-type: none"> • Servomotor de puerta. • Sensor de movimiento PIR 	La puerta se abre cuando el sensor de movimiento detecta presencia y la puerta se cierra cuando no detecta ninguna presencia.
Encender y apagar Buzzer	<ul style="list-style-type: none"> • Fococelda. • Buzzer. 	El Buzzer se enciende cuando detecta una cantidad de Luz muy alta y el buzzer se apaga cuando no detecta gran intensidad de Luz.
Encender y apagar Ventilador	<ul style="list-style-type: none"> • Fococelda. • Ventilador. 	El Ventilador se enciende cuando detecta una cantidad de luz muy alta y el ventilador se apaga cuando no detecta gran intensidad de luz.
Imprimir en LCD	<ul style="list-style-type: none"> • LCD. 	Cuando se inicializa el Arduino UNO, este imprime un mensaje estático en el LCD.

Software

Una vez que ya se ha abordado totalmente el hardware, y además que ya se conoce las acciones que puede realizar la casa domótica, es de gran importancia abordar el software y como se implementó cada uno de los elementos para poder alcanzar con los objetivos propuestos para el proyecto, además de ayudar a alcanzar en su totalidad el producto final.

Programación de la casa domótica

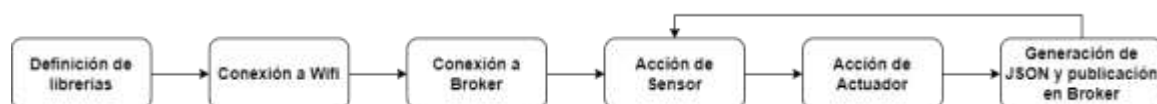
Como ya se ha mencionado, el controlador de la casa domótica es un módulo ESP8266 y para el LCD en específico es un Arduino UNO. El lenguaje de programación es propio de esta plataforma y para poder programarlo se usa su propio IDE y un computador.

Para entender y resumir el funcionamiento del programa del controlador, se puede observar el diagrama de la figura 38, donde se ve el proceso desde que se inicializa el módulo como tal hasta que se publica en un tópico del bróker y después el

proceso se repite, desde el inicio de la acción, ya que todas las secciones de configuración no vuelven a ocurrir.

Figura 38

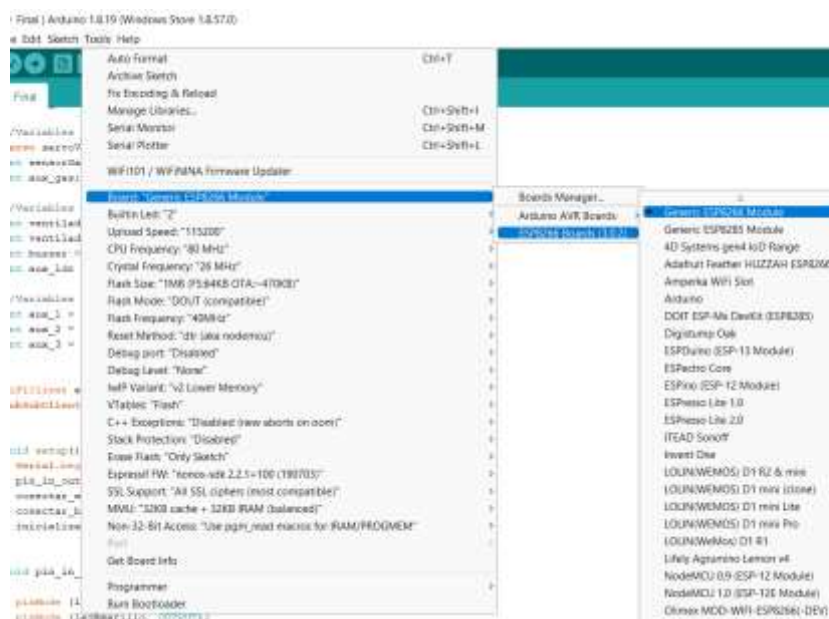
Funcionamiento del programa de Arduino.



Conexión con Arduino. Para poder conectarse con Arduino se debe instalar el IDE de desarrollo, una vez instalado, se debe seleccionar la placa, en este caso Esp8266 o Arduino UNO, y finalmente el puerto, en la Figura 39 se puede ver la interfaz propia y como se ven las opciones. Finalmente, para cargar el programa se debe seleccionar compilar y cargar.

Figura 39

Arduino IDE y conexión.



Estructura y funcionamiento del programa de la casa domótica.

El programa desarrollado en Arduino tiene una estructura secuencial, esto quiere decir que va siguiendo las líneas de código en el orden en el que están escritas. Es por ello que es importante ir definiendo una a una las secciones más relevantes del programa.

Librerías. Esta es la parte inicial de cualquier programa desarrollado en Arduino y para el propósito del proyecto las utilizadas fueron las necesarias para conectarse a una red WiFi, conectarse a un Broker, generar archivos JSON y poder manejar los servomotores por PWM y estas son:

```
<ESP8266WiFi.h>
<PubSubClient.h>
<ArduinoJson.h>
<Servo.h>
```

Conexión a una red WiFi. Esta es la parte fundamental si se quiere reconocer y conectarse a una red WiFi, ya que se definen dos atributos importantes, que son el nombre de la red y la clave de la red, esto se logra con las siguientes líneas de código:

```
const char *ssid = "...";
const char *password = "...";
```

Conexión al Broker. Así como para el Wifi, esta es la parte fundamental si se quiere reconocer y conectarse a un Broker de mensajería, primero los atributos que se deben definir, son la IP y el puerto de la Red donde se encuentra desplegado, y después se puede definir los tópicos que se desee tanto para publicar o suscribirse, esto se logra con las siguientes líneas de código:

```
const char *mqtt_broker = "...";
```

```
const int mqtt_port = ...;

const char *topic = "...";
```

Inicialización de variables y definición de pines. Es importante en todo programa de Arduino, definir tanto el número de GPIO al que pertenece cada sensor o actuador y el modo en el funcionará este PIN, es así que se lo puede establecer como entrada o salida, o si va a estar en estado de HIGH o LOW de ser el caso, esto se logra con las siguientes líneas de código:

```
int ledBlanco = 16;

pinMode (ledBlanco, OUTPUT);

digitalWrite (ledBlanco, LOW);
```

Generación del mensaje JSON y publicación en el Broker. Si bien un Broker funciona con JSON por defecto, no está demás generar un archivo o un mensaje en este formato, para tener una mejor versatilidad al momento de querer como manejar los datos y finalmente este mensaje sea enviado al Broker con un mensaje de publicación, obviamente siempre que sea después de alguna acción predeterminada que genere la publicación. Esto se hace con las siguientes líneas de código:

```
void json () {

    StaticJsonDocument<256> doc;

    doc["..."] = "...";

    doc["..."] = "...";

    doc["..."] = "...";

    char out[128];

    serializeJson(doc, out);

    client.publish(topic, out);

}
```

Una vez se ha abordado la estructura y funcionamiento general del programa de Arduino, es importante saber cuáles son los tópicos en los cuales el ESP8266 publica y además saber cuál es el actuador que corresponde a los mismos, en la tabla 13 se puede ver este resumen.

Tabla 13

Actuador, acción y tópico del broker.

Actuador	Acción	Tópico de Broker
Luces Led	<ul style="list-style-type: none"> • Encendidas • Apagadas 	lucesLed
Servomotor Puerta	<ul style="list-style-type: none"> • Abierta • Cerrada 	puerta
Buzzer	<ul style="list-style-type: none"> • Encendido • Apagado 	buzzer
Servomotor Ventana	<ul style="list-style-type: none"> • Encendido • Apagado 	ventana
Ventilador	<ul style="list-style-type: none"> • Encendido • Apagado 	ventilador

Frontend – Aplicación de Verificación

Como se explicó en el capítulo anterior, y en las figuras de la 16 a la 19, la aplicación debe tener una estructura y diseño apropiados. De este modo, la implementación de la aplicación de verificación tiene dos partes principales en cuanto a su desarrollo, la parte estética de forma y la parte funcional. Para la primera, se utiliza HTML y CSS, estructura y diseño respectivamente. El código no va a profundizarse en su totalidad, si se quiere ahondarlo, puede encontrarse en los Anexos.

Por otro lado, la parte de la funcionalidad si es muy importante de abordar, ya que, con ella es la que se configura las acciones necesarias para cumplir con los objetivos del proyecto, para su implementación se utiliza JavaScript, este lenguaje de programación ayuda a darle funcionalidad a los componentes previamente desarrollados en HTML y CSS, la característica más importante es que JavaScript permite que se generen peticiones HTTP que pueden enviarse al Backend y así consumir los servicios REST del mismo. De igual manera, todo el código se encuentra en los anexos.

Configuraciones de la aplicación de verificación. Solo se abordará el código programado en JavaScript, Como ya se mencionó, pues se tiene varios botones que son encargados de la comunicación con Backend, sin embargo, todos son similares y el tipo de petición HTTP es la misma, de tipo GET, en la figura 40 se puede ver todos los controles de la aplicación, que cumplen la misma función, pero para diferente actuador o componente ciber físico de que se quiere consultar ya sea su estado actual o su historial.

Figura 40

Controles de la aplicación de verificación.



Una vez se tiene todos los controles en mente, se tiene que mencionar que su configuración es la misma, con la diferencia de que cambian las URLs a donde se envía la petición HTTP y además la gestión de datos con las respuestas también varía, ya que se recibirán datos diferentes, el código es el siguiente:

```
let url = "http://localhost:8090/api";
var xmlhttp = new XMLHttpRequest();

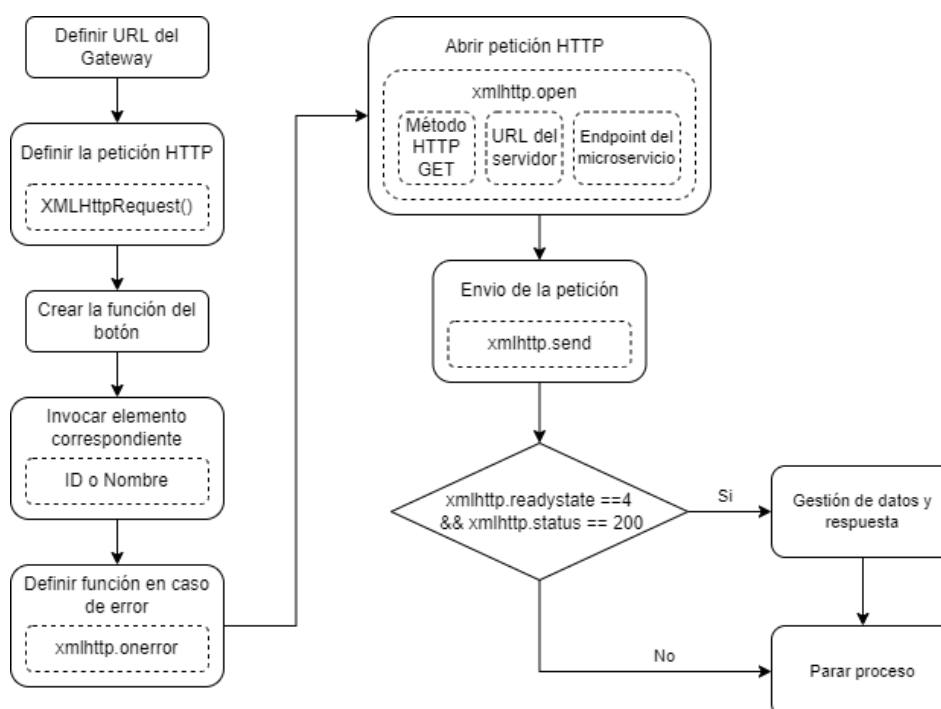
function do_actual_ledblanco() {
  document.getElementById("boton_actual_ledblanco").style.backgroundColor = "#abd9f5";
  let api = "/lucesled/actual";
  xmlhttp.timeout = 1000;
  xmlhttp.onerror = function () {
    swal({
      title: "Atención!",
      text: "No se estableció conexión.",
      icon: "warning",
    });
  };

  xmlhttp.open("GET", url + api);
  xmlhttp.send();
  xmlhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
      respuesta = JSON.parse(xmlhttp.responseText);
      json = respuesta.estadoActuador;
      if (json == "Encendido") {
        swal({
          title: "Encendido",
          text: "Led Blanco Encendido.",
          icon: "success",
        });
      } else {
        swal({
          title: "Apagado",
          text: "Led Blanco Apagado.",
          icon: "error",
        });
      }
    }
  };
};
```

Para entender mejor su funcionamiento, se tiene la figura 41, un diagrama de flujo donde en primer lugar se define la URL del Gateway al que se conectara la aplicación, después se crea la petición HTTP con la clase XMLHttpRequest(), después se crea la función, se usa como ejemplo la función do_actual_ledblanco(), que hace referencia al botón del estado actual del led blanco, a continuación se llama el botón por el ID definido en el HTML, a continuación se abre la petición con xmlhttp.open(), donde se termina de definir la url con el endpoint específico del servicio del componente ciber físico al que se quiere acceder, con xmlhttp.send() se hace el envío de la petición y después se hace la validación que depende del xmlhttp.readystate que debe ser 4 y del xmlhttp.status que debe ser igual a 200 para ser correcto y proceder a validar la respuesta, y entonces dependiendo de lo que se obtenga en dicha respuesta se notificará al usuario con la alerta que corresponda.

Figura 41

Diagrama de flujo de las peticiones HTTP de la aplicación de verificación.



Finalmente, una vez explicado el funcionamiento de las peticiones HTTP desde el Frontend, es necesario conocer cuáles son las rutas de los microservicios de los componentes ciberfísicos a las que van dirigidas dichas peticiones para poder conocer que es lo que se podría esperar de respuesta.

Tabla 14

Botones de la aplicación de verificación y sus atributos.

Control de verificación	Ruta HTTP	Endpoint del microservicio del componente ciberfísico	Mensaje recibido en JSON
Luz Led Blanca			
Estado actual	http://localhost:8090/api	/lucesled/actual	Encendido
Historial	http://localhost:8090/api	/lucesled/historial	Apagado Historial de acciones
Luz Led Amarilla			
Estado actual	http://localhost:8090/api	/lucesled/actual	Encendido
Historial	http://localhost:8090/api	/lucesled/historial	Apagado Historial de acciones
Puerta			
Estado actual	http://localhost:8090/api	/puerta/actual	Abierto
Historial	http://localhost:8090/api	/puerta/historial	Cerrado Historial de acciones
Ventana			
Estado actual	http://localhost:8090/api	/ventana/actual	Abierto
Historial	http://localhost:8090/api	/ventana/historial	Cerrado Historial de acciones
Ventilador			
Estado actual	http://localhost:8090/api	/ventilador/actual	Activado
Historial	http://localhost:8090/api	/ventilador/historial	Desactivado Historial de acciones
Buzzer			
Estado actual	http://localhost:8090/api	/buzzer/actual	Activado
Historial	http://localhost:8090/api	/buzzer/historial	Desactivado Historial de acciones

Para utilizar la aplicación de verificación, se necesita un navegador Web, puede ser cualquiera, después lo que se hace es abrir el archivo index.html como se puede ver en la figura 42.

Figura 42

Archivo principal de la aplicación de verificación

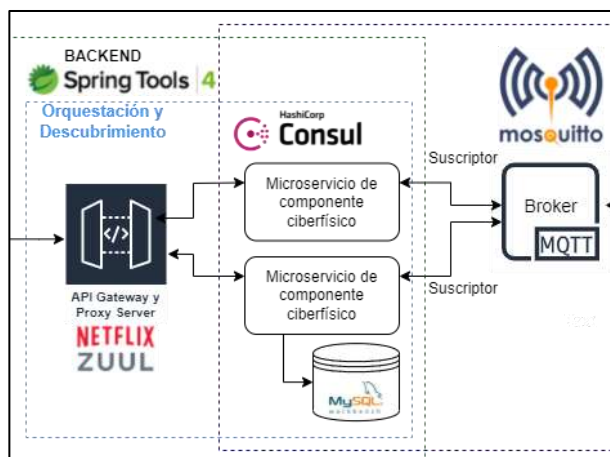
CSS	1/28/2022 12:52 PM	File folder	
IMG	1/28/2022 2:58 PM	File folder	
JS	1/28/2022 12:52 PM	File folder	
index.html	1/31/2022 8:03 PM	Chrome HTML Do...	7 KB

Backend – Arquitectura del servidor

Una vez se ha explicado en los capítulos y las secciones anteriores, se tiene la arquitectura general un poco más desarrollada, en la figura 43 se puede ver la arquitectura del Backend finalizada, junto con el resto de tecnologías y herramientas importantes. A partir de esto se puede ir desglosando las partes importantes de la implementación.

Figura 43

Estructura final del backend.

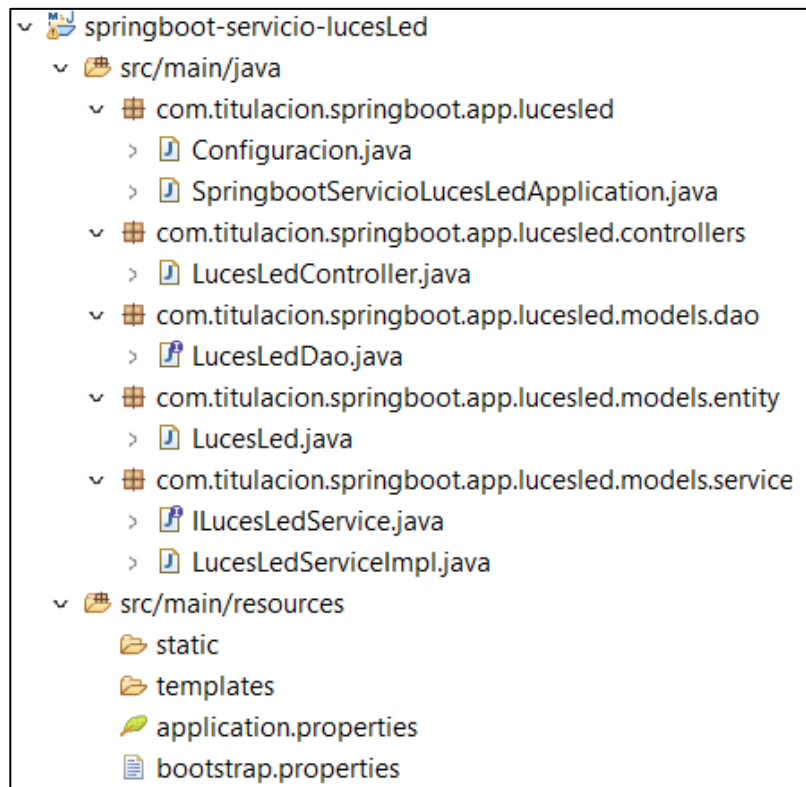


Es claro que el Backend es el punto central y el encargado de toda la gestión de los datos, se utilizan microservicios que están relacionados directamente con los componentes ciberfísicos, se dispone de un servidor que cumple el rol de Proxy y de Gateway y finalmente se tiene al agente de orquestación y descubrimiento, que es el pilar fundamental para que toda la arquitectura del Backend funcione. Como framework de desarrollo se escogió Spring Tools 4 por su versatilidad al momento de utilizar varios tipos de configuración y protocolos, como es el caso del trabajo. Junto con las anotaciones de Spring y Java, permite tener prestaciones muy amplias, pudiendo tener un sistema tanto con HTTP como con MQTT.

Estructura de los microservicios. Los microservicios son los encargados de recibir las peticiones por parte del Frontend y los de recibir la información por parte de la casa domótica y guardar estos datos en las bases de datos respectivas. La distribución de los microservicios es la siguiente: El paquete principal se divide en tres subpaquetes: aplicación (nombre del microservicio), controladores (controllers) y modelos (models), este último se divide en otros tres subpaquetes: objeto de acceso a datos (dao - data access object), entidad (entity) y servicio (service). Dentro de cada uno de los paquetes mencionados se encuentran diferentes clases importantes para que todo funcione bien y adicionalmente existe el paquete de recursos y dentro se encuentran el archivo de propiedades de la aplicación (application.properties) y las propiedades bootstrap (bootstrap.properties). Como ejemplo se utiliza el microservicio correspondiente al componente ciber físico de las luces led y en la figura 44 se puede ver la estructura general que tiene el mismo, con todos los paquetes y las clases importantes.

Figura 44

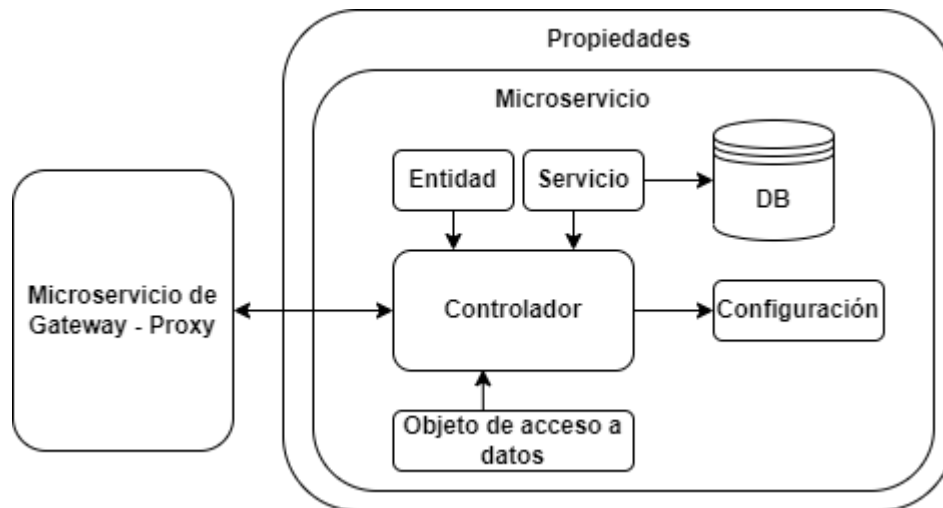
Estructura de un microservicio de un componente ciber físico.



Es importante mencionar cual es la relación que tienen entre todos los microservicios, esto ya se abordó en el capítulo de diseño, pero en la figura 45 se puede ver como cada uno de los paquetes se relaciona directamente con el controlador, adicionalmente es importante tener en cuenta que para poder acceder a cada uno de los microservicios se tiene otros microservicios como el del Gateway que funciona también como Proxy y de este modo, mediante las rutas que se le proporcione puede acceder sin ningún problema a cada uno de los endpoints que se desee.

Figura 45

Relación entre microservicios.



Configuración de las clases del microservicio. El funcionamiento de los microservicios depende de las clases que se encuentran en cada paquete, es por ello que es importante ir definiendo en qué consisten cada una, se usará como ejemplo el microservicio *LucesLed* como ejemplo para poder ir explicando cada una de las clases importantes, así como sus métodos.

Clases del paquete de la aplicación. La primera clase que se tiene en este paquete es llamada *SpringbootServicioLucesLedApplication*, su código completo se encuentra en los Anexos, internamente no cuenta con métodos, pero cuenta con la siguiente estructura:

```

@SpringBootApplication
public class SpringbootServicioLucesLedApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringbootServicioLucesLedApplication.class, args);
    }
}
  
```

Es una clase donde se tiene una anotación como habilitadora, la cual es:

```
@SpringBootApplication
```


Junto con la clase anterior, se tiene la clase *Configuracion*, esta clase define todos los atributos importantes que debe tener el proyecto, como por ejemplo los atributos de REST y las suscripciones a los tópicos más importantes y que se van a utilizar del Broker, cada una de las anotaciones están relacionadas con métodos importantes como:

```
@Bean
public MqttPahoClientFactory mqttClientFactory() {
    DefaultMqttPahoClientFactory factory = new DefaultMqttPahoClientFactory();
    MqttConnectOptions options = new MqttConnectOptions();
    options.setServerURIs(new String[] { "tcp://localhost:1883" });
    options.setCleanSession(true);
    factory.setConnectionOptions(options);
    return factory;
}
```

En este método se define la dirección IP donde se encuentra desplegado el bróker de mensajería, como se puede ver en este caso se trata de la red local, en el puerto 1883, adicionalmente se puede definir otras opciones, como el usuario o una contraseña, en el caso de que sean necesarias.

El siguiente método donde se define el nombre del cliente que se suscribirá al broker, es el siguiente:

```
@Bean
public MessageProducer inbound() {
    MqttPahoMessageDrivenChannelAdapter adapter = new
    MqttPahoMessageDrivenChannelAdapter("ServerInLucesLed",
        mqttClientFactory(), "#");
    adapter.setCompletionTimeout(5000);
    adapter.setConverter(new DefaultPahoMessageConverter());
    adapter.setQos(2);
    adapter.setOutputChannel(mqttInputChannel());
    return adapter;
}
```

El nombre del suscriptor es *ServerInLucesLed*, adicionalmente se puede establecer el timeout de las respuestas, y el nivel de calidad de servicio que se necesite, finalmente esto se establece como adaptador y se despliega.

Por último, se tiene el método donde se define el canal de entrada y el tópicos específico donde se va a recibir los mensajes de este microservicio en el broker.

```
@Bean
@ServiceActivator(inputChannel = "mqttInputChannel")
public MessageHandler handler() {
    return new MessageHandler() {
        @Override
        public void handleMessage(Message<?> message) throws MessagingException
        {
            String topic =
            message.getHeaders().get(MqttHeaders.RECEIVED_TOPIC).toString();
            if(topic.equals("lucosLed")) {
                System.out.println("Topic:"+topic);
                String jsonString = (String) message.getPayload();
                Gson g = new Gson();
                LucosLed l = g.fromJson(jsonString, LucosLed.class);
                lucosLedService.save(l);
            }
        }
    };
}
```

El tópicos específico para este microservicio es *"lucosLed"*, entonces una vez que se reciba un mensaje en el broker, se hace una validación de si se trata del tópicos correspondiente y entonces la cadena de caracteres se le transforma a un árbol de datos tipo JSON, esto se lo convierte a la entidad respectiva del microservicio, que es *LucosLed* y finalmente se guardan estos datos en la base de datos a la que este conectada el microservicio.

En la figura 46 se puede apreciar como se ve el paquete de la aplicación, con sus ambas clases, sus anotaciones y sobre todo los métodos y funciones más importantes.

Figura 46

Estructura del paquete de la aplicación de un microservicio.



Clases del paquete del controlador. La única clase que se tiene dentro de este paquete es la llamada *LucesLedController*, es la clase encargada de gestionar todas las peticiones HTTP de tipo GET que llegan desde el Frontend y además tiene la función de tener un endpoint el cual está orientado a que el agente de descubrimiento pueda verificar si el microservicio se encuentra estable o no. El código puede encontrarse en los Anexos. Pero los métodos más importantes son:

```

@GetMapping("/historial")
public List<LucesLed> detalleLocal(){
    return lucesLedService.findAll();
}

@GetMapping("/actual")
public LucesLed detalleUltimo(){
    return lucesLedService.encontrar();
}
  
```

En estos métodos claramente se puede ver los respectivos endpoints “/historial” y “/actual”, que cumplen el rol de responder la petición de tipo GET que se solicite, además utiliza los métodos declarados en la interfaz y declaración del servicio, como son findAll y encontrar. También se tiene otro método importante, encargado de enviar las respuestas del “Health Check” que el agente de descubrimiento y orquestación necesita.

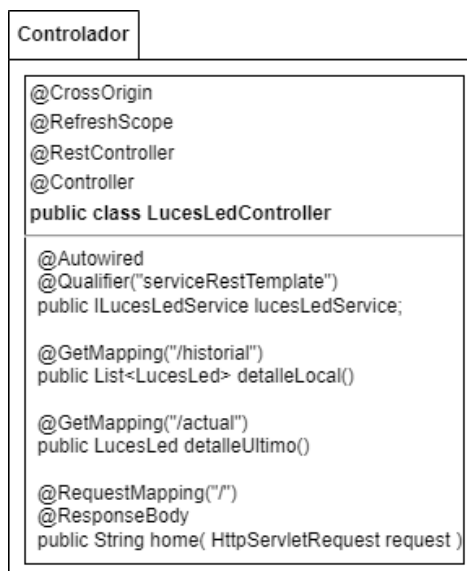
```
@RequestMapping("/")
@ResponseBody
public String home( HttpServletRequest request ) {
    StringBuilder sb=new StringBuilder();
    sb.append( "<h1>Servicio Luces Led</h1>" );
    return sb.toString();
}
```

Internamente el método puede estar compuesto por cualquier contenido, ya sea un String, o algún tipo de dato, lo importante es que se encuentre habilitado.

En la figura 47 se puede ver la estructura del paquete controlador, donde constan las anotaciones y métodos importantes.

Figura 47

Estructura del paquete del controlador de un microservicio.



Clases del paquete DAO. La clase que se encuentra en este paquete es la

llamada *LucesLedDao*, que tiene la siguiente estructura:

```
@Repository
public interface LucesLedDao extends CrudRepository<LucesLed, Long>{
    LucesLed findTopByOrderByIdDesc();
}
```

Esta clase está orientada para que se puedan utilizar las prestaciones del repositorio que se defina, en este caso CRUD, que significa CREAR, LEER, ACTUALIZAR y BORRAR, esto se hereda del repositorio con:

extends CrudRepository

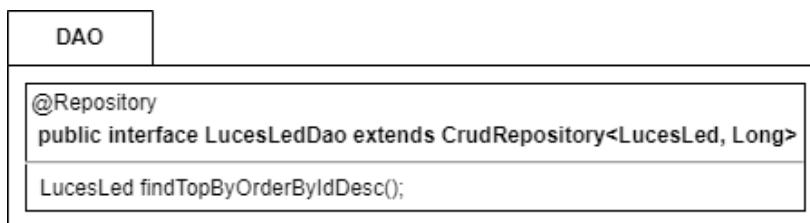
Además, también se puede definir funcionalidades propias, como por ejemplo el único método implementado:

findTopByOrderByIdDesc();

Este método está relacionado con la búsqueda específica del último elemento en la base de datos en la figura 48 puede verse su estructura y su código se encuentra en los Anexos.

Figura 48

Estructura del paquete DAO de un microservicio.



Clases del paquete de entidad. La clase de este paquete está definido para definir las variables con las cuales se podrá distinguir el componente ciber físico de los demás, la estructura de esta clase es la siguiente:

```
@Entity
@Table(name="luces_led")
public class LucesLed {
```

```

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
private String idSensor;
private String estadoSensor;
private String idActuador;
private String estadoActuador;
private String hora =
LocalDateTime.now().truncatedTo(ChronoUnit.SECONDS).format(DateTimeFormatter.ISO_DATE_TIME);
}

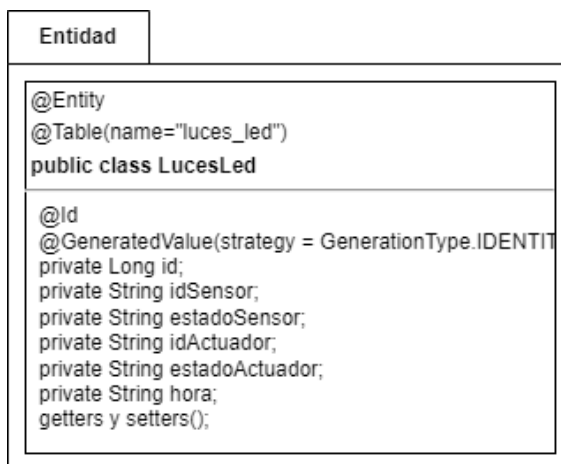
```

En este caso tienen los atributos de: *id*, *idSensor*, *estadoSensor*, *idActuador*, *estadoActuador*, *hora*, que significan literalmente lo mismo que su variable definida, es por todo esto que a la clase se define como entidad, ya que es única y los objetos que se creen a partir de esta clase, son propios de este microservicio, aquí se definen las variables que se desea guardar en la base de datos, además se puede definir de los métodos getters y setters de cada variable, en caso de que se necesite utilizarlos en otra clase del proyecto, su código completo se encuentra en los Anexos.

En la figura 49 se puede observar la estructura del paquete de la entidad.

Figura 49

Estructura del paquete de entidad de un microservicio.



Clases del paquete del servicio. En este paquete hay dos clases

ILucesLedService y *LucesLedServiceImpl*, ambas funcionan juntas, ya que una es la interfaz y la otra es su implementación, los métodos definidos en ambas cumplen el rol de enviar, recibir y guardar los datos, su código completo puede encontrarse en los Anexos. La clase de la interfaz tiene la siguiente estructura:

```
public interface ILucesLedService {
    public List<LucesLed> findAll();
    public LucesLed save(LucesLed lucesLed);
    public LucesLed encontrar();
}
```

La clase de implementación tiene la siguiente estructura:

```
@Service("serviceRestTemplate")
public class LucesLedServiceImpl implements ILucesLedService {

    @Autowired
    private LucesLedDao lucesLedDao;

    @Override
    public List<LucesLed> findAll() {
        return (List<LucesLed>) lucesLedDao.findAll();
    }

    @Override
    @Transactional
    public LucesLed save(LucesLed lucesLed) {
        return lucesLedDao.save(lucesLed);
    }

    @Override
    public LucesLed encontrar() {
        return lucesLedDao.findTopByOrderByIdDesc();
    }
}
```

Dentro de ambas clases se definen y se sobrescriben los mismos métodos, esto es para facilitar la comunicación de los datos y para utilizar al máximo las prestaciones que brinda Spring para el desarrollo de este tipo de aplicaciones y servicios, los métodos son los siguientes:

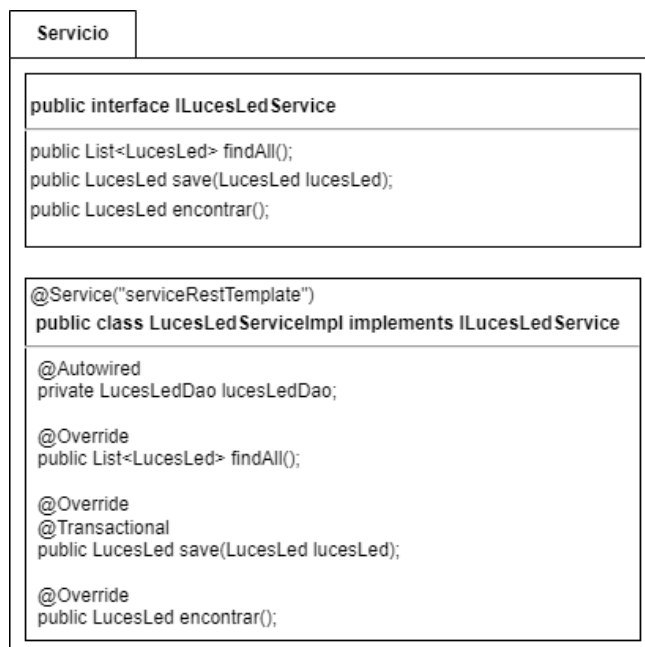
- **findAll():** Método relacionado directamente con el repositorio CRUD, literalmente hace lo que su etiqueta significa “encontrar todos”, con este método se devuelve todos los elementos de la base de datos.
- **save():** Método encargado de guardar cada elemento relacionado con la variable o la entidad que se le envíe como atributo.
- **encontrar():** Método encargado de encontrar un elemento en específico, dependiendo de las variables que se le asignen como atributo, este método utiliza el método definido en el DAO manualmente:

“findTopByOrderByIdDesc();”

En la figura 50 se puede ver la estructura del paquete del servicio.

Figura 50

Estructura del paquete del servicio de un microservicio.



Archivos del paquete de recursos. Los archivos de este paquete son los más importante en cuanto a configuración de los microservicios. El archivo de recursos tiene la siguiente estructura, además de otras ciertas líneas de código. Su código completo se encuentra en los Anexos.

```
spring.application.name=servicio-lucesled
server.port=${PORT:0}
```

```
spring.cloud.consul.host=127.0.0.1
spring.cloud.consul.port=8500
```

El archivo bootstrap tiene la siguiente estructura:

```
spring.application.name=servicio-lucesled
management.endpoints.web.exposure.include=*
```

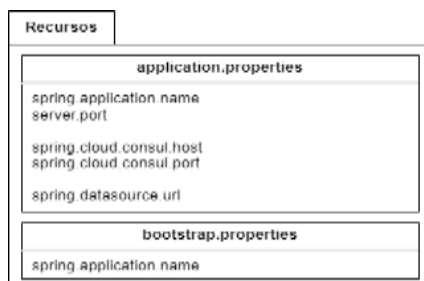
Cada línea es de fácil comprensión, pero el rol que cumplen algunas líneas, son los siguientes:

- **spring.application.name:** El nombre de la aplicación o del microservicio.
- **server.port:** El puerto que utilizará el microservicio.
- **spring.cloud.consul.host:** La dirección IP del agente de descubrimiento.

Tanto en el archivo de propiedades, como en el archivo bootstrap se debe definir el nombre del microservicio al que se está haciendo referencia, en este caso es el *lucesled*. En la figura 50 se puede apreciar la estructura de estos archivos, y sobre todo las líneas de código más importantes.

Figura 51

Estructura del paquete de recursos de un microservicio.

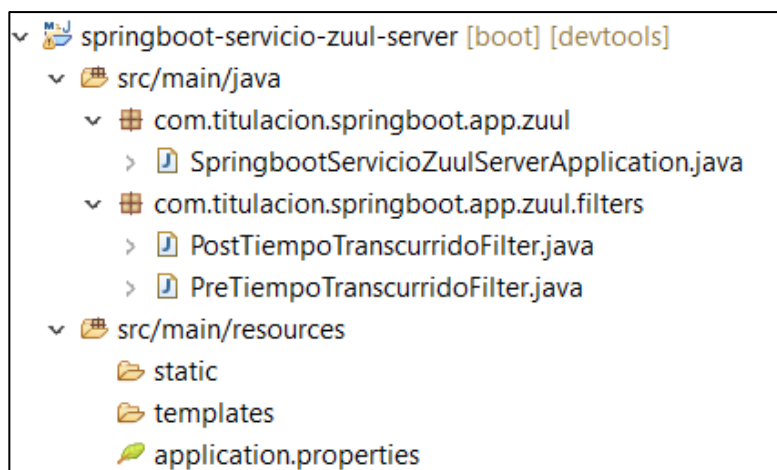


Microservicios extra. Además de los microservicios que representan cada uno de los componentes ciberfísicos, también se tiene un microservicio especial que cumple el rol de Gateway y de Proxy, este se llama Zuul.

Estructura del microservicio Zuul. Este microservicio cumple el rol de Gateway y Proxy dentro de la arquitectura, su estructura puede verse en la figura 52, tiene tres paquetes importantes, el de aplicación, el de los filtros y el de recursos.

Figura 52

Estructura del microservicio Zuul



Clases del microservicio Zuul. Este microservicio tiene algunas clases y archivos, pero sin duda los más importantes son los archivos dentro del paquete de recursos, donde se definen todos sus atributos, pero sobre todo las rutas de los microservicios de los componentes ciberfísicos que se quiere alcanzar mediante las peticiones HTTP del frontend, por ejemplo, la ruta del microservicio lucesled, se vería de la siguiente manera:

```

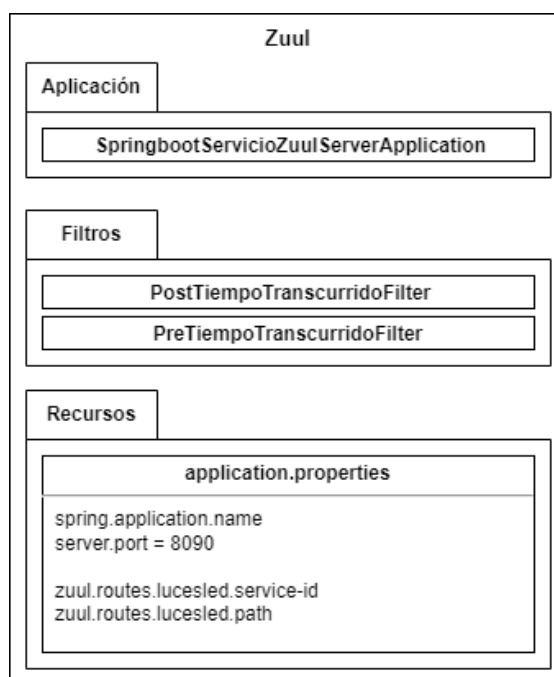
zuul.routes.lucesled.service-id=servicio-lucesled
zuul.routes.lucesled.path=/api/lucesled/**

```

Se tiene tanto el id del servicio, como la ruta en sí, es importante mencionar que el puerto por defecto de Zuul es el 8090. El código completo se encuentra en los Anexos. Y en la figura 53 se puede ver cómo están conformadas las clases de este microservicio.

Figura 53

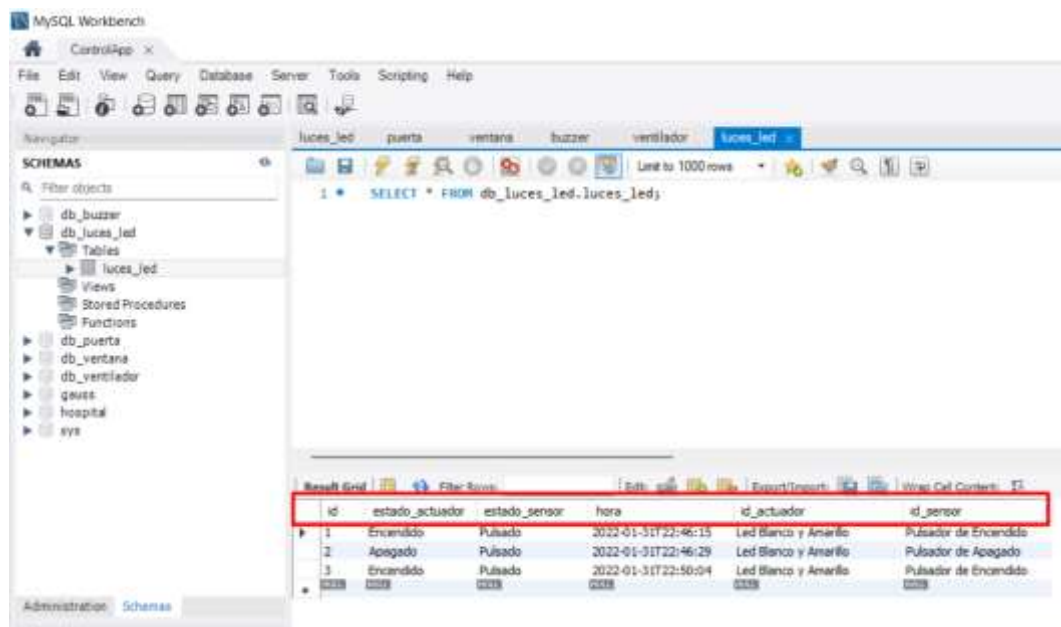
Clases del servicio Zuul



Estructura de las bases de datos de los microservicios. Estas son bases de datos relacionales, y funcionan mediante SQL, la herramienta para la gestión de las mismas en el proyecto es MySQL Workbench, donde toda su configuración puede realizarse usando su interfaz. En la figura 53 puede apreciarse como se ve la interfaz de Workbench y como ejemplo se utiliza la base de datos y la tabla del microservicio *lucesled*. Los valores a guardar dependerán de los requerimientos, pero para este caso son: id, estado_actuador, estado_sensor, hora, id_sensor e id_actuador.

Figura 54

Estructura de la base de datos de un microservicio.



Agente orquestador y de descubrimiento. Para implementar el agente que es encargado de descubrir y orquestar en primer lugar se debe descargarlo desde la página oficial, la cual es la de Hashicorp Consul. El funcionamiento del agente consiste en identificar los microservicios, en este caso los que están relacionados con los componentes ciberfísicos, una vez identificados, su siguiente función consiste en orquestar las acciones, es decir va indicando cual es el microservicio que debe realizar la petición sin importar ningún orden, su responsabilidad es recopilar o entregar información acerca de los mismos, esto lo realiza una vez verifique si el microservicio se encuentra en línea, gracias a una función propia que se llama "Health Check" que es similar a ir enviando "pings" en una red lo que indica que se encuentra activo el o los componentes ciberfísicos en este caso.

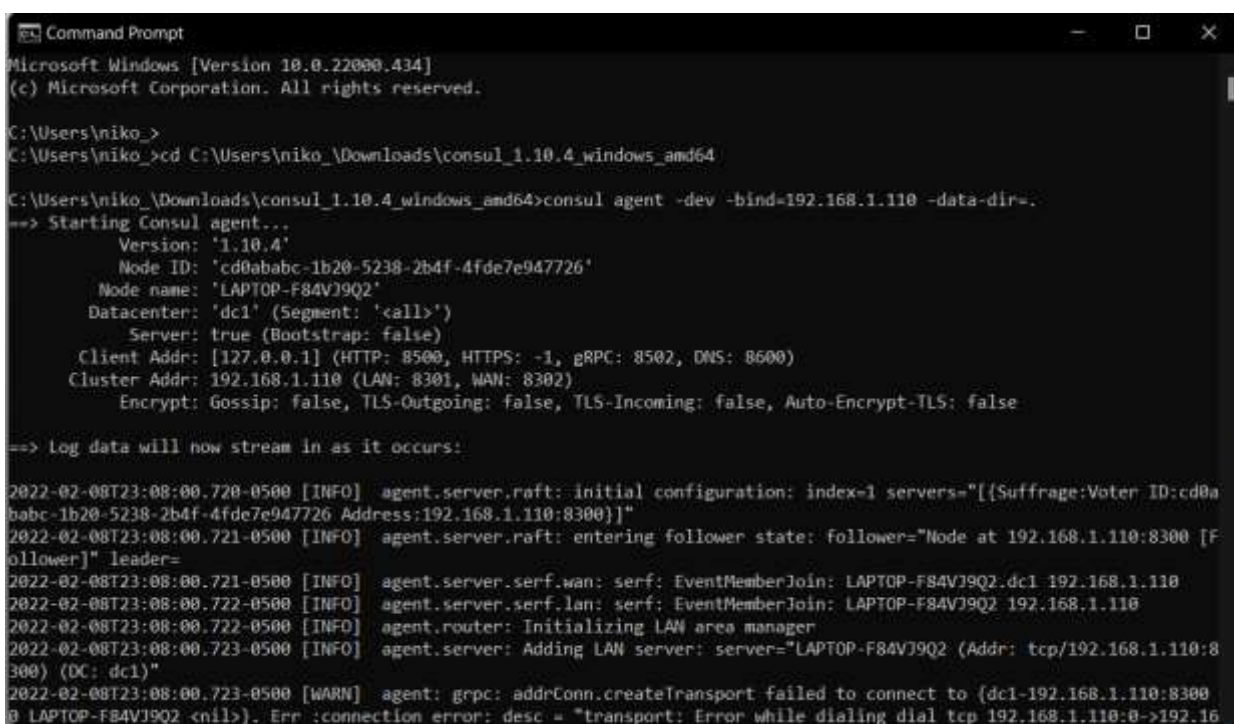
Es importante saber cuáles son los pasos a seguir para poder hacer que corra en la maquina local, para esto debemos ingresar a la carpeta donde este guardado, una vez ahí, se debe ingresar el siguiente comando:

```
consul agent -dev -bind=192.168.1.110 -data-dir=.
```

En dicho comando, deben constar la IP de la maquina en la que va a correr junto con otros comandos importantes, de este modo como se aprecia en la figura 55, el agente empezará a revidar uno a uno los microservicios de los componentes ciberfísicos, para saber si se encuentran levantados y en buen estado.

Figura 55

Despliegue del agente Consul.



```

Command Prompt
Microsoft Windows [Version 10.0.22000.434]
(c) Microsoft Corporation. All rights reserved.

C:\Users\niko_>
C:\Users\niko_>cd C:\Users\niko_Downloads\consul_1.10.4_windows_amd64
C:\Users\niko_Downloads\consul_1.10.4_windows_amd64>consul agent -dev -bind=192.168.1.110 -data-dir=.
==> Starting Consul agent...
    Version: '1.10.4'
    Node ID: 'cd0ababc-1b20-5238-2b4f-4fde7e947726'
    Node name: 'LAPTOP-F84VJ9Q2'
    Datacenter: 'dc1' (Segment: 'all')
    Server: true (Bootstrap: false)
    Client Addr: [127.0.0.1] (HTTP: 8500, HTTPS: -1, gRPC: 8502, DNS: 8600)
    Cluster Addr: 192.168.1.110 (LAN: 8301, WAN: 8302)
    Encrypt: Gossip: false, TLS-Outgoing: false, TLS-Incoming: false, Auto-Encrypt-TLS: false

==> Log data will now stream in as it occurs:

2022-02-08T23:08:00.720-0500 [INFO] agent.server.raft: initial configuration: index=1 servers="[{"Suffrage:Voter ID:cd0ababc-1b20-5238-2b4f-4fde7e947726 Address:192.168.1.110:8300}]"
2022-02-08T23:08:00.721-0500 [INFO] agent.server.raft: entering follower state: follower="Node at 192.168.1.110:8300 [Follower]" leader=
2022-02-08T23:08:00.721-0500 [INFO] agent.server.serf.wan: serf: EventMemberJoin: LAPTOP-F84VJ9Q2.dc1 192.168.1.110
2022-02-08T23:08:00.722-0500 [INFO] agent.server.serf.lan: serf: EventMemberJoin: LAPTOP-F84VJ9Q2 192.168.1.110
2022-02-08T23:08:00.722-0500 [INFO] agent.router: Initializing LAN area manager
2022-02-08T23:08:00.723-0500 [INFO] agent.server: Adding LAN server: server="LAPTOP-F84VJ9Q2 (Addr: tcp/192.168.1.110:8300) (DC: dc1)"
2022-02-08T23:08:00.723-0500 [WARN] agent: grpc: addrConn.createTransport failed to connect to {dc1-192.168.1.110:8300 @ LAPTOP-F84VJ9Q2 <nil>}. Err :connection error: desc = "transport: Error while dialing dial tcp 192.168.1.110:8300->192.168.1.110:8300: connect: connection refused"

```

Una vez el agente se encuentra desplegado, se puede acceder a la interfaz local en donde se puede verificar de manera mucho más visual todos los microservicios que se encuentran en ese momento habilitados, como se puede ver en la figura 56, y si se

quiere poder ver a más detalle uno de los microservicios en específico se puede acceder al mismo y ver su información, sobre todo el health check que es lo que indica que el servicio se encuentra funcionando de manera óptima, como se puede ver en la figura 57.

Figura 56

Interfaz del agente de orquestación y descubrimiento.

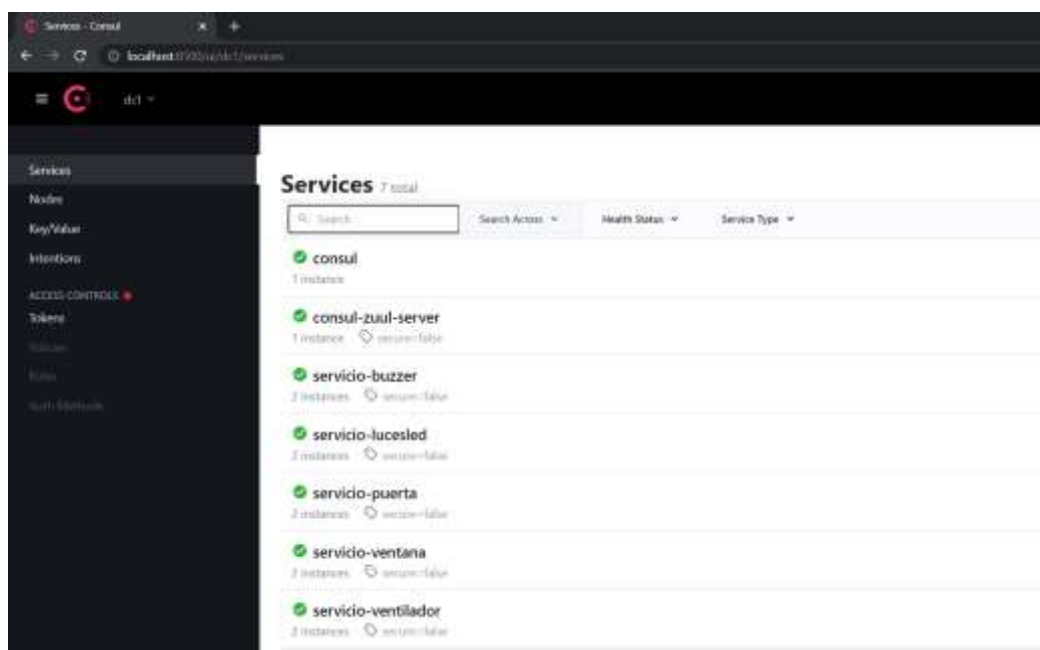


Figura 57

Estado de un microservicio en el agente de descubrimiento.



La importancia del agente de orquestación y descubrimiento es que permite identificar a los microservicios relacionados con los componentes ciberfísicos independientemente de la ubicación donde se encuentren levantados, de su dirección IP y su puerto, esto gracias a que se maneja por identificadores, además permite tener varias instancias del mismo microservicio funcionando a la par, esto con el objetivo de que si alguna falla, se tenga un respaldo en la ruta que los datos pueden seguir, finalmente haciendo buen uso de su interfaz se puede tener toda la información necesaria de todos los nodos que se encuentren implementados, lo más importante es la notificación de “Health Check”, que es lo que indica que el microservicios se encuentra en línea o si tiene algún error.

Broker MQTT. El último elemento importante por definir es el Broker, su funcionamiento es ser un gestor de mensajes y además permite tener tópicos específicos donde cada componente puede suscribirse y publicar dependiendo de los requerimientos, en la figura 58 se puede ver el despliegue del broker en el puerto 1883 y como se va verificando si los suscriptores se encuentran activos mediante los PINGREQ y PINGRES. Para levantar el Broker en la computadora local, se debe descargarlo, acceder a la carpeta donde se lo instala e ingresar la siguiente línea de comando:

```
mosquitto -c mosquitto.conf -v
```

Figura 58

Despliegue y funcionamiento del Broker.

```
C:\Program Files\mosquitto>
C:\Program Files\mosquitto>mosquitto -c mosquitto.conf -v
1644382195: mosquitto version 2.0.14 starting
1644382195: Config loaded from mosquitto.conf.
1644382195: Opening ipv6 listen socket on port 1883.
1644382195: Opening ipv4 listen socket on port 1883.
1644382195: mosquitto version 2.0.14 running
1644382221: New connection from 127.0.0.1:60979 on port 1883.
1644382222: New client connected from 127.0.0.1:60979 as serverInBuzzer (p2, c1, k60).
1644382222: No will message specified.
1644382222: Sending CONNACK to serverInBuzzer (0, 0)
1644382222: Received SUBSCRIBE from serverInBuzzer
1644382222:     # (QoS 2)
1644382222: serverInBuzzer 2 #
1644382222: Sending SUBACK to serverInBuzzer
1644382231: New connection from 127.0.0.1:61040 on port 1883.
1644382232: New client connected from 127.0.0.1:61040 as serverInLucesLed (p2, c1, k60).
1644382232: No will message specified.
1644382232: Sending CONNACK to serverInLucesLed (0, 0)
1644382232: Received SUBSCRIBE from serverInLucesLed
1644382232:     # (QoS 2)
1644382232: serverInLucesLed 2 #
1644382232: Sending SUBACK to serverInLucesLed
1644382234: New connection from 127.0.0.1:61059 on port 1883.
1644382234: New client connected from 127.0.0.1:61059 as serverInPuerta (p2, c1, k60).
1644382234: No will message specified.
1644382234: Sending CONNACK to serverInPuerta (0, 0)
1644382234: Received SUBSCRIBE from serverInPuerta
1644382234:     # (QoS 2)
1644382234: serverInPuerta 2 #
1644382234: Sending SUBACK to serverInPuerta
1644382235: New connection from 127.0.0.1:61079 on port 1883.
1644382236: New client connected from 127.0.0.1:61079 as serverInVentana (p2, c1, k60).
1644382236: No will message specified.
1644382236: Sending CONNACK to serverInVentana (0, 0)
1644382236: Received SUBSCRIBE from serverInVentana
```


Capítulo V

Pruebas de Validación

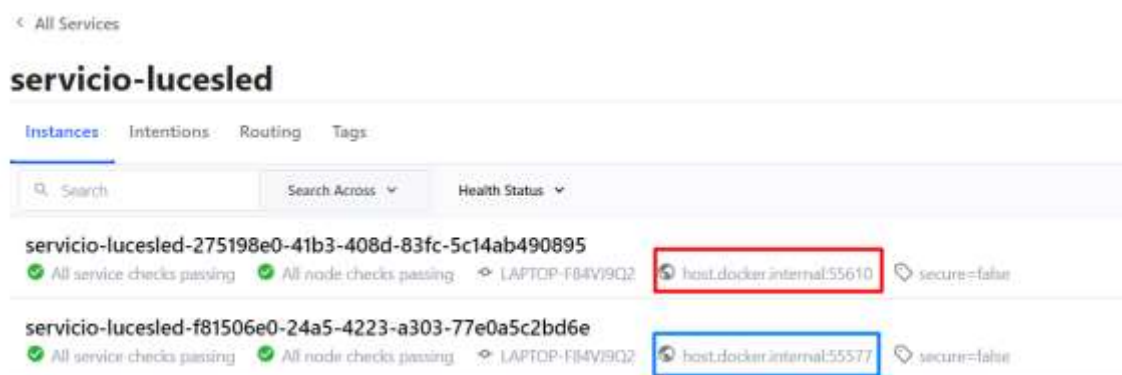
Pruebas de funcionamiento

Una vez el sistema se encuentra implementado y funcionando en su totalidad, se procede a realizar las pruebas de funcionamiento, es de este modo que se logró ir probando cada una de las prestaciones establecidas. Primero se tiene que poder comprobar que en caso de que una instancia de un microservicio falle, otra tome su lugar y se pueda seguir ocupando el sistema sin problema, para esto se utilizará como ejemplo al microservicio de las lucesled.

Para empezar, se tiene que desplegar dos instancias del mismo microservicio, como se puede ver en la figura 59, la una está funcionando en el puerto 55610 y la otra en el 55577.

Figura 59

Instancias del microservicio.



Para saber que datos se están enviando, se procede a publicar en el broker de forma manual, utilizando la ventana de comandos, en la figura 60 se puede ver la línea de código que se utiliza:

Figura 60

Publicación en el broker.



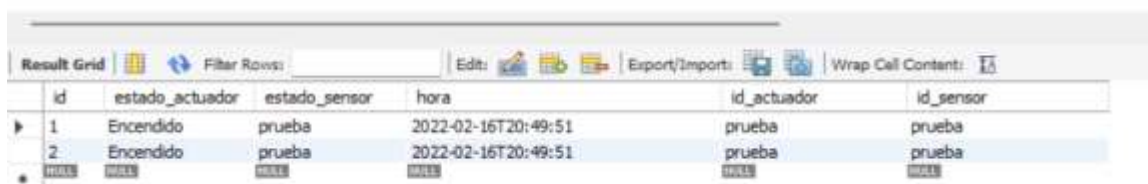
```

C:\Program Files\mosquitto>mosquitto_pub -t lucesLed -m '{"idSensor":"prueba","estadoSensor":"prueba","idActuador":"prueba","estadoActuador":"Encendido"}'
C:\Program Files\mosquitto>
  
```

Entonces, al tener dos instancias del mismo microservicio, y ambas en funcionamiento, en la base de datos se verán dos registros iguales, en la figura 61 se puede ver la base de datos del microservicio:

Figura 61

Base de datos del microservicio.

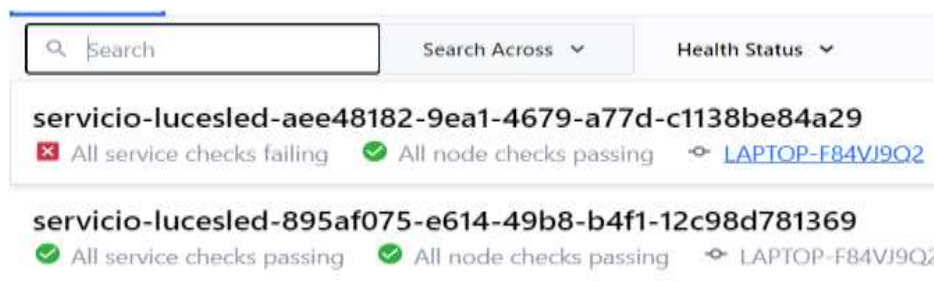


id	estado_actuador	estado_sensor	hora	id_actuador	id_sensor
1	Encendido	prueba	2022-02-16T20:49:51	prueba	prueba
2	Encendido	prueba	2022-02-16T20:49:51	prueba	prueba

Para probar que el agente de descubrimiento está funcionando bien, se procede a detener una de las instancias del microservicio, en la figura 62 se puede ver como solo una de las instancias se encuentra activa y el resto no.

Figura 62

Instancia activa del microservicio.



Search	Search Across	Health Status
servicio-lucesled-ae48182-9ea1-4679-a77d-c1138be84a29	All service checks failing	All node checks passing
servicio-lucesled-895af075-e614-49b8-b4f1-12c98d781369	All service checks passing	All node checks passing

De este modo se vuelve a publicar en el broker y en la base de datos solo va a constar un registro, que corresponde a la única instancia del microservicio activa, esto puede verse en la figura 63.

Figura 63

Registro de una instancia en la base de datos.



id	estado_actuador	estado_sensor	hora	id_actuador	id_sensor
1	Encendido	prueba	2022-02-16T21:13:55	prueba	prueba

En el Frontend en la aplicación de control siempre va a constar el último registro, por lo que en este caso tomando en cuenta el estado del actuador “Encendido”, se visualizará dicha notificación, como en la figura 66. La ventaja de poder manejar múltiples instancias, es que el agente de orquestación y descubrimiento siempre va a seleccionar la mejor para el envío y recepción de datos, y en caso de que alguna se encuentre caída, se tiene una o algunas otras de respaldo, es así que se garantiza el funcionamiento del sistema, independientemente de la ubicación física, el nombre de la computadora o la dirección ip de la red donde se encuentren levantadas las instancias de los microservicios.

Una vez se logró probar el funcionamiento de las instancias de los microservicios, y como el agente de orquestación y descubrimiento es capaz de gestionarlas y en caso de que alguna falle, usar una de respaldo, se debe probar el funcionamiento general del sistema, incluyendo la casa domótica. El funcionamiento de sistema, en primer lugar, consiste en activar un sensor, el cual va a activar un actuador, en la siguiente imagen se puede ver como se activa el pulsador de encendido de las luces led y las luces led se encienden:

Figura 64

Pulsador de encendido de las luces led.



Al estar las luces led encendidas, entonces los datos de esta acción del actuador, pasan a enviarse al broker, donde se guardan en una base de datos correspondiente a este componente ciberfísicos, los datos en la base de datos pueden verse como en la siguiente figura, donde constan todas las variables definidas en la entidad, que son: estado del actuador, estado del sensor, la fecha y hora, la id del actuador y la id del sensor.

Figura 65

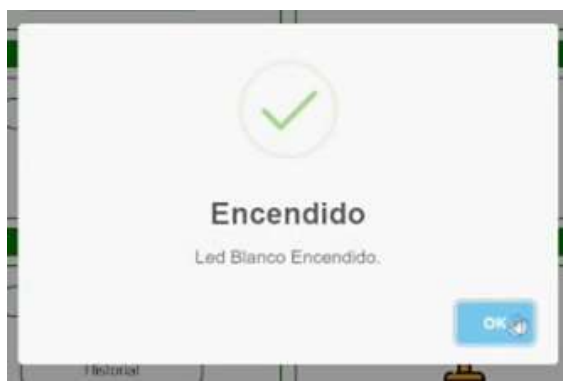
Base de datos.

id	estado_actuador	estado_sensor	hora	id_actuador	id_sensor
1	Encendido	Pulsado	2022-01-31T22:08:43	Led Blanco y Amarillo	Pulsador de Encendido
2	Encendido	Pulsado	2022-01-31T22:08:47	Led Blanco y Amarillo	Pulsador de Encendido
3	Apagado	Pulsado	2022-01-31T22:08:59	Led Blanco y Amarillo	Pulsador de Apagado
4	Encendido	Pulsado	2022-01-31T22:09:42	Led Blanco y Amarillo	Pulsador de Encendido
5	Apagado	Pulsado	2022-01-31T22:10:01	Led Blanco y Amarillo	Pulsador de Apagado
6	Encendido	Pulsado	2022-01-31T22:12:19	Led Blanco y Amarillo	Pulsador de Encendido

Una vez se encuentran los datos guardados, la aplicación de verificación puede acceder a los mismos utilizando los controles implementados, de este modo si se quiere verificar el estado actual de las luces, como en este caso están encendidas, la notificación se verá de la siguiente manera:

Figura 66

Estado actual del actuador



También se tiene la opción de verificar el historial de acciones que ha realizado el actuador, de este modo, si se escoge esa opción, la notificación se verá de la siguiente manera:

Figura 67

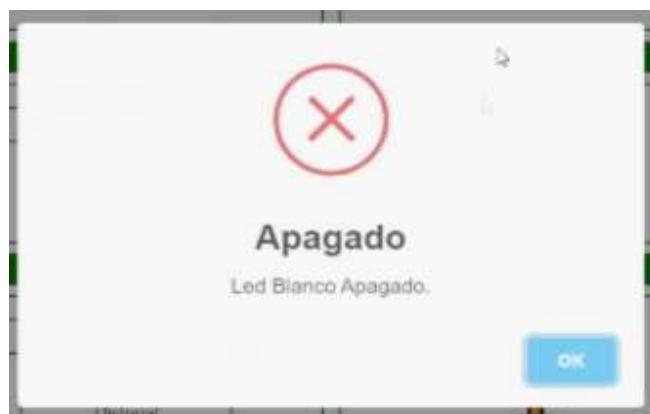
Historial de las acciones del actuador.



En el caso de que el estado actual del actuador sea apagado, la notificación se verá de la siguiente manera:

Figura 68

Estado apagado del actuador.



Finalmente, en el caso de que la aplicación de verificación no encuentre una conexión estable con los servicios del Backend, entonces se mostrará la notificación de fallo, la cual se verá de la siguiente manera:

Figura 69

Notificación de fallo de conexión.



Pruebas de carga

Además de las pruebas con el hardware, es importante tener constancia de que la arquitectura de software está funcionando bien. Una vez la arquitectura se encuentra completamente implementada y desplegada, se tiene que utilizar un script de Gatling, que funciona con Scala, un lenguaje de programación específico para este tipo de pruebas, entonces se tiene que configurar todas las peticiones que son posibles de enviar y que el sistema pueda recibir, de este modo se obtiene un total de 12 peticiones las cuales corresponden a los 12 botones que se tiene en la aplicación de verificación, una vez configuradas todas las peticiones, se tiene que correr el Script, y entonces se obtiene los resultados que se pueden ver en la figura 70, donde se ve que el 100% de las peticiones fueron exitosas, además se puede ver que su tiempo de respuesta es menor a 800ms, en la figura 71 se puede ver las 12 peticiones con algunos otros datos importantes.

El código de Gatling tiene tres secciones importantes, la primera es donde se define el tipo de protocolo, HTTP en este caso y todos sus atributos importantes, como el encabezado y el tipo de datos:

```
HttpProtocolBuilder httpProtocol =  
    http  
        // Here is the root for all relative URLs  
        .baseUrl("http://localhost:8090")  
        .acceptHeader("text/html,application/xhtml+xml,application/xml;q  
=0.9,*/*;q=0.8")
```

La segunda parte importante es el escenario donde se definen las peticiones y el tipo, en este caso son de tipo GET, además se le asigna un nombre y a cada una de las mismas y si se desea se puede poner pausas entre las mismas, en tiempo de ejecución:

```

ScenarioBuilder scn =
    scenario("Prueba de funcionamiento")
        .exec(http("Luz blanca actual").get("/api/lucesled/actual"))
        .pause(1)
        .exec(http("Luz blanca
historial").get("/api/lucesled/historial"))
        .pause(1)

```

Por último, se tiene la configuración final para la ejecución, donde se definen la cantidad de usuarios que se quiere simular, en el tiempo que se quiere realizar y el protocolo que se utilizará para la prueba, de este modo se tiene la opción de probar la cantidad de usuarios que se desee, siempre y cuando se acerque a la realidad:

```

setUp(scn.injectOpen(rampUsers(1000).during(60)).protocols(httpProtocol));

```

Figura 70

Prueba de funcionamiento de las peticiones y tiempo de respuesta.



Nota: Obtenida de los datos generados por Gatling.

Figura 71

Peticiones de la prueba de funcionamiento.

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	12	12	0	0%	0.657	8	26	148	312	467	505	104	135
Luz blanca actual	1	1	0	0%	0.071	505	505	505	505	505	505	505	0
Luz blan...istorial	1	1	0	0%	0.071	32	32	32	32	32	32	32	0
Luz amarilla actual	1	1	0	0%	0.071	10	10	10	10	10	10	10	0
Luz amar...istorial	1	1	0	0%	0.071	8	8	8	8	8	8	8	0
Puerta actual	1	1	0	0%	0.071	146	146	146	146	146	146	146	0
Puerta historial	1	1	0	0%	0.071	26	26	26	26	26	26	26	0
Ventana actual	1	1	0	0%	0.071	150	150	150	150	150	150	150	0
Ventana historial	1	1	0	0%	0.071	24	24	24	24	24	24	24	0
Ventilador actual	1	1	0	0%	0.071	155	155	155	155	155	155	155	0
Ventilad...istorial	1	1	0	0%	0.071	25	25	25	25	25	25	25	0
Buzzer actual	1	1	0	0%	0.071	147	147	147	147	147	147	147	0
Buzzer historial	1	1	0	0%	0.071	22	22	22	22	22	22	22	0

Nota: Obtenida de los datos generados por Gatling.

Para otras pruebas de carga, de igual manera se utiliza Gatling, debido a la facilidad de configurar cantidad de usuarios, número de peticiones y tiempo, entonces de este modo se tendrá 10 peticiones para 100, 200, 400, 800, 1000 y 5000 usuarios en un tiempo de 60 segundos.

Prueba de carga con 100 usuarios.

En la figura 72, se puede ver la cantidad de peticiones totales, donde es importante notar que el 100% de las peticiones se completaron, además en la figura 73 se puede ver la máxima cantidad de peticiones por segundo, y la gráfica del comportamiento de las peticiones en total.

Figura 72

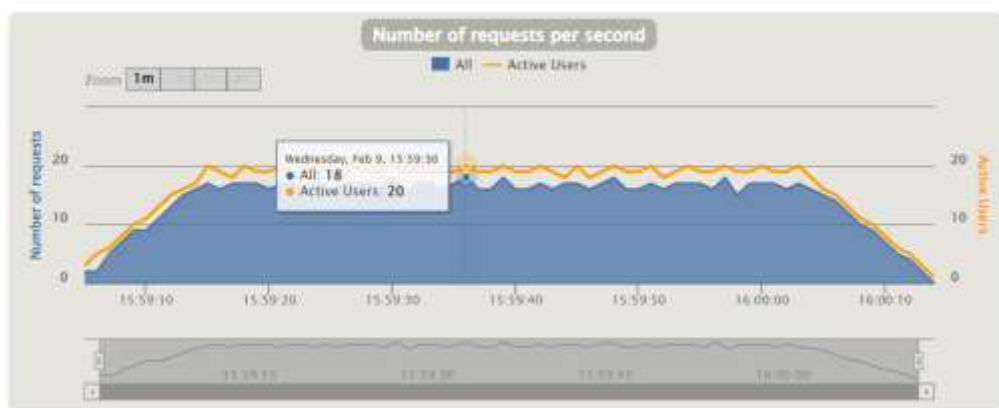
Prueba de carga con 100 usuarios.



Nota: Obtenida de los datos generados por Gatling.

Figura 73

Pico de peticiones con 100 usuarios.



Nota: Obtenida de los datos generados por Gatling.

Prueba de carga con 200 usuarios

En la figura 74, se puede ver la cantidad de peticiones totales, donde es importante notar que el 100% de las peticiones se completaron, además en la figura 75 se puede ver la máxima cantidad de peticiones por segundo, y la gráfica del comportamiento de las peticiones en total.

Figura 74

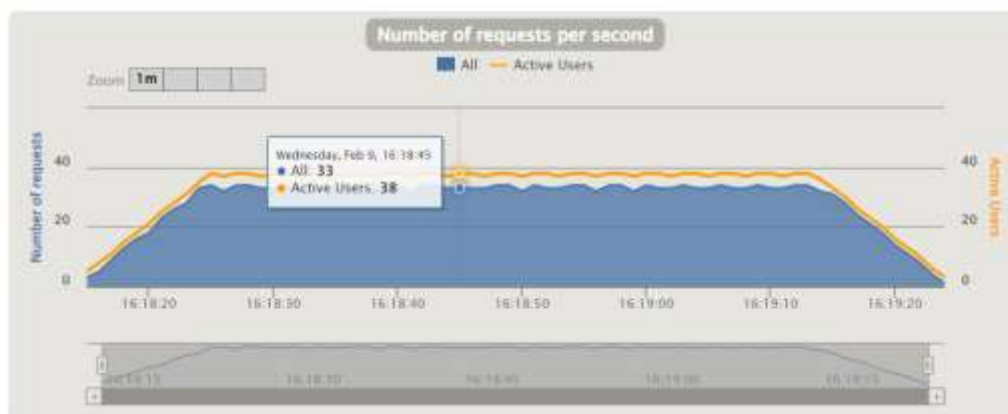
Prueba de carga con 200 usuarios.



Nota: Obtenida de los datos generados por Gatling.

Figura 75

Pico de peticiones con 200 usuarios.



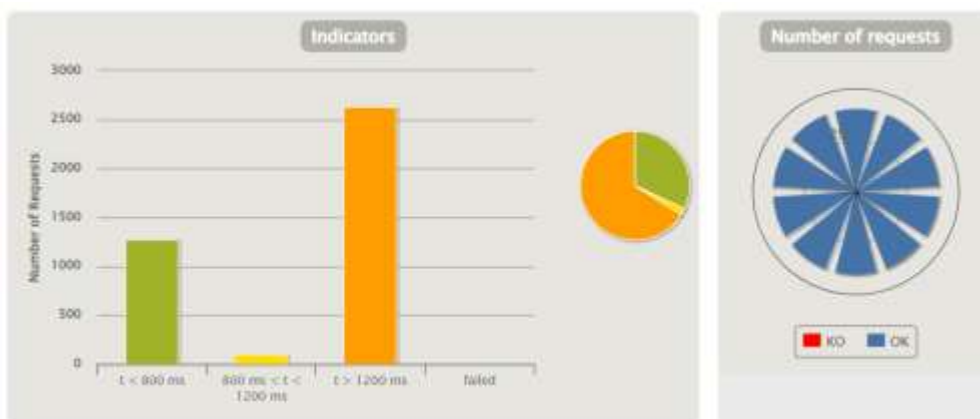
Nota: Obtenida de los datos generados por Gatling.

Prueba de carga con 400 usuarios

En la figura 76, se puede ver la cantidad de peticiones totales, donde es importante notar que el 32% de las peticiones se completaron, sin embargo, con un tiempo de respuesta mayor, además en la figura 77 se puede ver la máxima cantidad de peticiones por segundo, y la gráfica del comportamiento de las peticiones en total.

Figura 76

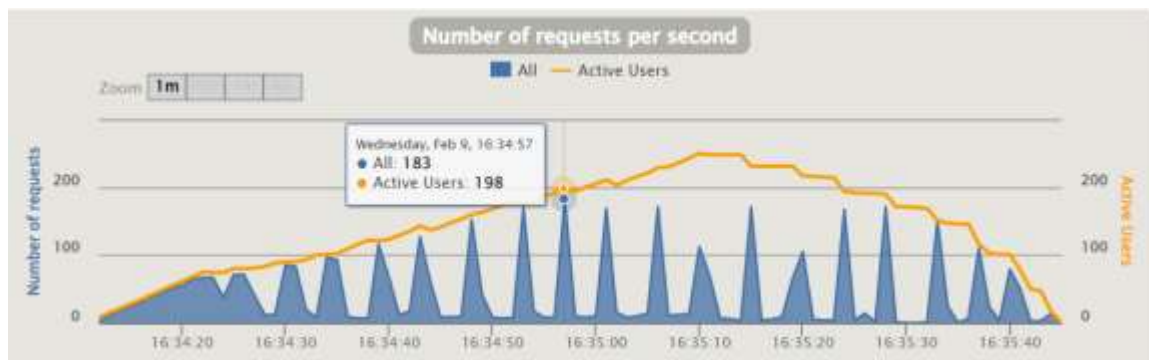
Prueba de carga con 400 usuarios.



Nota: Obtenida de los datos generados por Gatling.

Figura 77

Pico de peticiones para 400 usuarios.



Nota: Obtenida de los datos generados por Gatling.

Prueba de carga con 800 usuarios

En la figura 78, se puede ver la cantidad de peticiones totales, donde es importante notar que el 100% de las peticiones se completaron, sin embargo, con un tiempo de respuesta mayor, además en la figura 79 se puede ver la máxima cantidad de peticiones por segundo, y la gráfica del comportamiento de las peticiones en total.

Figura 78

Prueba de carga con 800 usuarios.



Nota: Obtenida de los datos generados por Gatling.

Figura 79

Pico de peticiones para 800 usuarios.



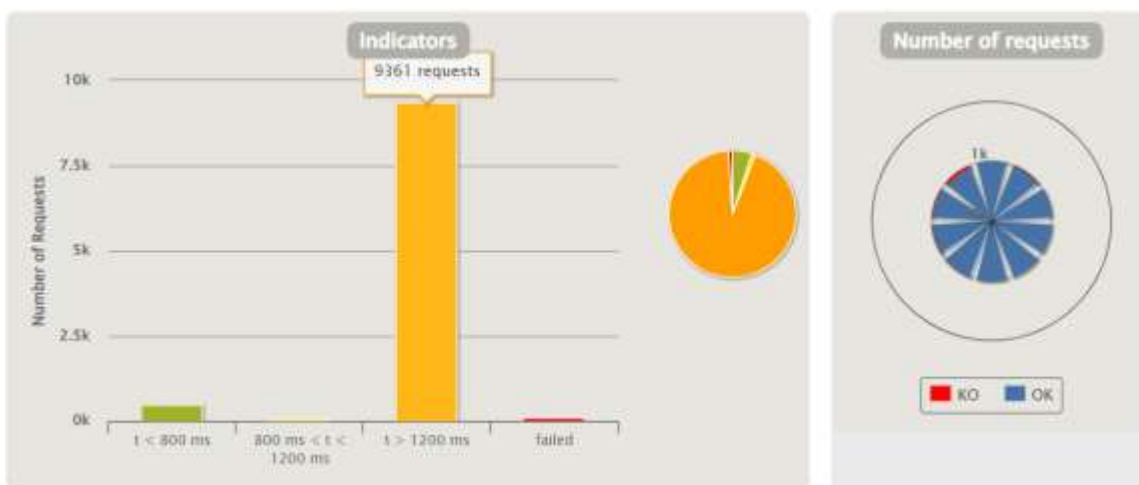
Nota: Obtenida de los datos generados por Gatling.

Prueba de carga con 1000 usuarios

En la figura 80, se puede ver la cantidad de peticiones totales, donde es importante notar que el 99% de las peticiones se completaron, sin embargo, con un tiempo de respuesta mayor, además en la figura 81 se puede ver la máxima cantidad de peticiones por segundo, y la gráfica del comportamiento de las peticiones en total.

Figura 80

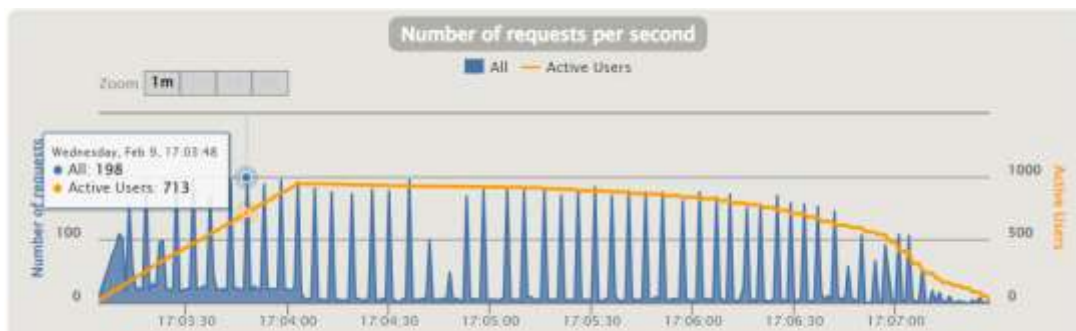
Prueba de carga con 1000 usuarios.



Nota: Obtenida de los datos generados por Gatling.

Figura 81

Pico de peticiones para 1000 usuarios.



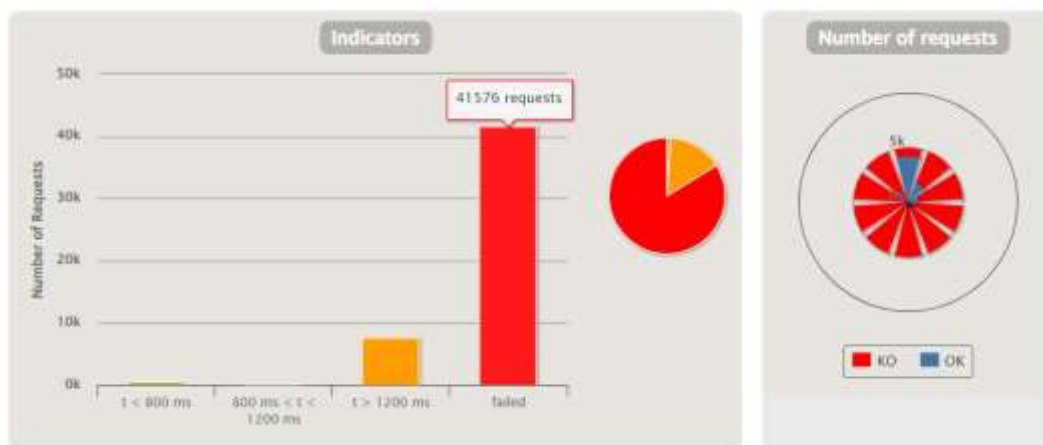
Nota: Obtenida de los datos generados por Gatling.

Prueba de carga con 5000 usuarios

En la figura 82, se puede ver la cantidad de peticiones totales, donde es importante notar que el 98.72% de las peticiones fallaron, además en la figura 83 se puede ver la máxima cantidad de peticiones por segundo, y la gráfica del comportamiento de las peticiones en total.

Figura 82

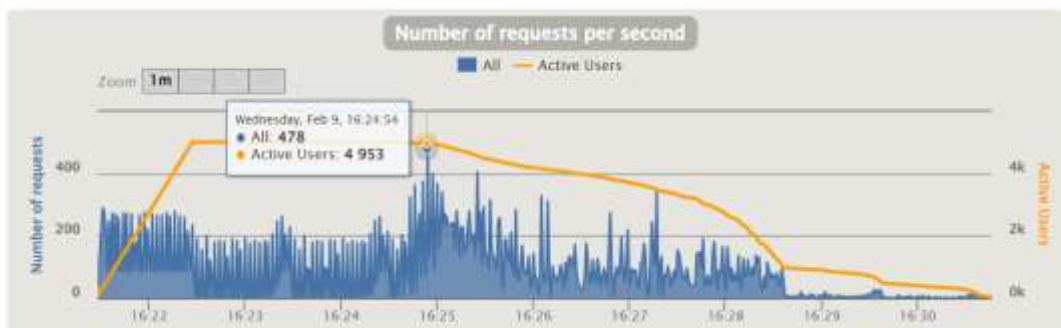
Prueba de carga con 5000 usuarios.



Nota: Obtenida de los datos generados por Gatling.

Figura 83

Pico de peticiones para 5000 usuarios.



Nota: Obtenida de los datos generados por Gatling.

Por último, las pruebas de carga cumplen con lo esperado, esto debido a que el backend funciona con un motor Apache Maven, donde en la documentación se indica que, si se sobrepasa las 160 peticiones por segundo, la eficiencia del sistema comienza a fallar, que es lo que se puede ver, a pesar de que las peticiones se completan en su totalidad, el tiempo de respuesta va aumentando, lo que es malo en ámbitos de QoS. En la tabla 15 se puede observar el resumen de los recopilados de las pruebas de carga y el comportamiento del sistema respecto a sus peticiones por segundo.

Tabla 15

Resumen de las pruebas de carga

Usuarios	Peticiones Totales	Máximas Pet/s	% Éxito	% Fallo	%Respuesta t < 800ms	%Respuesta t < 1200ms
100	1000	18	100%	0%	100%	0%
200	2000	34	100%	0%	100%	0%
400	4000	183	100%	0%	32%	68%
800	8000	199	100%	0%	7%	92%
1000	10000	198	99%	1%	5%	95%
5000	50000	478	1.28%	98.72%	1.28%	98.72%

Pruebas de usabilidad

Para estas pruebas se utiliza el sistema de escalas de usabilidad (System Usability Scale, SUS), esto con el fin de evaluar que tan amigable es todo el sistema con un usuario. Para esto se contó con 10 personas que pudieron usar la aplicación de manera individual, uno a uno, ya que se dispone solo de un sistema desplegado y únicamente de un dispositivo práctico, como es la casa domótica. Entonces para que las personas puedan probar el sistema completo, se las cito a cada una en horarios distintos para que evalúen cada una de las prestaciones que el sistema brinda y después se les pidió que llenen la encuesta donde constan las 10 preguntas de

evaluación que son importantes para poder sacar el puntaje SUS del sistema. Es importante mencionar, que dentro de las personas que pudieron probar el sistema, el 80% tiene preparación técnica y el 20% restante no la tenía, sin embargo, después de una breve explicación del funcionamiento del sistema, no tuvieron inconveniente al usar todos los elementos de la casa domótica y todos los controles de la aplicación de verificación. Es por estos motivos que se logró verificar y probar todas las funcionalidades de la casa domótica y se obtuvo los resultados expuestos en la tabla 16, las filas naranjas hacen referencia a los usuarios que no tienen preparación técnica, y el resto a los que sí, además ya se encuentran ponderadas todas las respuestas después de seguir las instrucciones propias del test para obtener el cálculo final de cada usuario. Después de hacer los cálculos y promediar, se obtiene el valor de 87.5/100, lo que lleva a la conclusión de que el sistema cumple con las expectativas y los objetivos planteados.

Tabla 16

Resultados del test de usabilidad.

Preguntas \ Usuarios	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Puntaje SUS
U1	5	2	4	1	4	1	5	1	5	1	92,5
U2	5	2	4	1	4	1	5	1	5	1	92,5
U3	3	1	5	1	5	1	5	5	5	1	85
U4	4	3	5	1	4	2	5	1	5	1	87,5
U5	4	5	5	1	5	1	5	1	5	1	87,5
U6	4	5	5	1	5	1	5	1	5	1	87,5
U7	5	1	5	1	5	1	5	5	5	1	90
U8	5	1	5	2	5	1	5	1	5	1	97,5
U9	5	2	5	2	5	1	5	1	5	1	95
U10	5	2	4	5	5	1	5	3	5	1	80
Promedio:											87,5

Por último, se puede mencionar que las respuestas de las dos personas sin conocimiento técnico presentan un promedio más bajo en comparación al resto y esto puede deberse a la percepción que cada uno de los usuarios interpreta según su manera de entender cada una de las preguntas y cada concepto al momento de evaluar y probar todo el sistema.

Capítulo VI

Conclusiones y Recomendaciones

Conclusiones

Para la comunicación entre el backend y la casa domótica se logró utilizar un broker de mensajería con el cual mediante el protocolo MQTT se tiene la ventaja de que el tiempo de respuesta sea menor, esto debido a que en el caso de MQTT todas y cada conexión que se establece, se mantiene abierta y se puede reutilizar siempre que se requiera una comunicación. Además, el broker puede tener la cantidad de tópicos que se desee para suscribirse y publicar en ellos, lo que facilita la gestión, manejo de los datos y además se ahorra el uso de recursos computacionales por su ligereza. Una de las desventajas de la utilización de un broker es que depende de que las conexiones de sus suscriptores se encuentren siempre activas, caso contrario pueden darse errores al enviar o recibir los datos, esto debido a que un broker no tiene la capacidad de resetear la conexión de un componente externo.

La utilización del microservicio Zuul y las librerías open source ofrecidas por Netflix, permite que se pueda acceder a cualquier microservicio de los componentes ciberfísicos sin necesidad de conocer su dirección local o su puerto, si no únicamente la dirección del Gateway, ya que es el que se encarga de redireccionar cualquier tipo de petición hacia donde corresponda, sin embargo, una desventaja es el alto consumo de los recursos computacionales que esto requiere.

Dentro del sistema se logró implementar la orquestación y descubrimiento de los componentes ciberfísicos utilizando el agente open source Consul de Hashicorp. Esta tecnología permite configurar cada microservicio relacionado a los componentes ciberfísicos para que puedan ser descubiertos y orquestados. El descubrimiento brinda gran fiabilidad en el sistema, además se logra identificar a los componentes

independientemente de su ubicación física, ya que, al configurarse identificadores propios de cada microservicio, se puede gestionarlos sin problema, además estos identificadores permiten orquestarlos de la manera que el sistema necesite. Las mayores ventajas de utilizar este tipo de tecnología son la facilidad que se tiene para escalar sistemas y su fácil operación, verificación y manejo.

La utilización de una arquitectura RESTful para la comunicación entre el frontend y el backend, permite que el usuario siempre obtenga una respuesta de acuerdo a su requerimiento, esto ya que, al basarse en el protocolo HTTP, siempre se espera una respuesta de confirmación o error, lo que permite hacer las validaciones necesarias para notificar al usuario que es lo que ha ocurrido. La aplicación de verificación en el Frontend es muy importante para poder comprobar la aplicabilidad práctica del proyecto, esto utilizando su interfaz visual e intuitiva, que dependiendo de los datos que se reciba a través de las peticiones HTTP de tipo GET muestran al usuario alertas y notificaciones claras que permiten verificar cualquier requerimiento.

Una vez realizada la prueba de usabilidad, y al obtener un puntaje de 87.5/100, se puede concluir que el sistema es amigable con el usuario, siempre y cuando se haga una explicación sencilla del funcionamiento de cada uno de los elementos que lo conforman, además esto es independiente de los conocimientos técnicos que los usuarios tengan.

Con los resultados obtenidos de las pruebas de carga, se concluye que en robustez, el sistema cumple con lo esperado, esto debido a que el backend funciona bajo un motor de compilación Apache Maven, que según sus especificaciones técnicas se sabe que si se sobrepasa las 160 peticiones por segundo, la eficiencia del sistema comienza a fallar, que es lo que se pudo evidenciar, a pesar de que en algunos casos

las peticiones logran completarse, los tiempos de respuesta van aumentando, lo que es bastante malo en ámbitos de QoS.

La utilización de un módulo ESP8266, permite que se pueda establecer una conexión a la red y al broker, sin embargo, solo puede utilizarse para el control de los componentes electrónicos, más no para alimentar los mismos, por lo que es importante contar con una fuente de alimentación externa tanto para los elementos como para el propio módulo, además dicha fuente tiene que suministrar una cantidad de corriente suficiente para que el sistema no llegue a fallar repentinamente.

Recomendaciones

El orden de despliegue del sistema es muy importante, ya que primero se necesita levantar el agente de orquestación y descubrimiento Consul, después el Broker MQTT, después todos los servicios del Backend y finalmente se debe encender y poner funcional la casa domótica.

Dentro de los microservicios, siempre se debe tener muy en cuenta que los nombres de cada uno de ellos deben corresponder a los que se encuentran en las configuraciones tanto del Gateway como del agente de orquestación y descubrimiento, caso contrario pueden darse errores.

Siempre se debe tener en cuenta las conexiones eléctricas de los componentes y sobre todo si se utiliza fuentes de alimentación distintas para los componentes, la tierra debe ser común, caso contrario el circuito nunca se cerrará.

Por último, si se quiere replicar este proyecto es importante contar con las versiones utilizadas de todas las herramientas y tecnologías, como la versión 4.9 RELEASE de STS 4, el JDK de Java de 1.8 o posterior, el Workbench 8.0 de SQL, JavaScript Vanilla, HTML 5 y CSS3, y Arduino IDE.

Trabajos Futuros.

Para los trabajos futuros se propone:

- Se puede ampliar la seguridad del sistema implementando interfaces de autenticación tanto en la aplicación de verificación como en la casa domótica, esto se lograría utilizando librerías especializadas como OAuth, esto brindaría mayor confiabilidad y seguridad para los usuarios y permitiría tener mayor personalización para cada usuario.
- Se puede utilizar herramientas y tecnologías diferentes, con el fin de medir el rendimiento de las mismas y tratar de tener un sistema lo más optimizado posible. Por ejemplo, en el caso del backend podría utilizarse un servicio en la nube como Amazon Web Services, en el caso del Frontend, frameworks específicos para el desarrollo de UIs como React, Angular, Vue, y para el caso de la casa domótica, puede utilizarse un microcontrolador más potente y robusto como es el caso de una Raspberry Pi.

Acrónimos

- CRUD: Crear, Leer, Actualizar, Eliminar.
- CPS: Sistema Ciber-Físico
- CSS: Hoja de estilos en cascada.
- DAO: Objeto de acceso a datos.
- HTML: Lenguaje de marco de hipertexto.
- HTTP: Protocolo de transferencia de hipertexto.
- IoT: Internet de las cosas.
- JS: JavaScript.
- MQTT: Protocolo de red ligero de publicación-suscripción.
- REST: Transferencia de estado representacional.
- SLR: Revisión sistemática de la literatura.
- SMS: Mapeo sistemático de la literatura.
- SOA: Arquitectura orientada a servicios.
- STS4: Spring Tools Suite 4.
- WoT: Web de las cosas.

Referencias

- Agarwal, K. (2019). Review and Performance Analysis on Wireless Smart Home and Home Automation using IoT. *Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 629-633.
- Agostinho, B. (2018). Smart Comm: A Smart Home Middleware Supporting Information Exchange. *44th Annual Conference of the IEEE Industrial Electronics Society*, 4678-4684.
- Alok, G., & Rahul, J. (2019). *IoT based electrical device surveillance and control system*. Ghaziabad,: IEEE.
- Alsuhaym, F., Al-Hadhrami, T., Saeed, F., & Awuson-David, K. (2021). Toward Home Automation: An IoT Based Home Automation System Control and Security. *IEEE-International Congress of Advanced Technology and Engineering (ICOTEN)*, 1-11.
- Alzaghoul, E. (2021, Febrero 12). *Wikimedia Inc*. Retrieved from Wikipedia la enciclopedia libre: https://en.wikipedia.org/wiki/Web_Services_Discovery
- Arjonilla, R. (2017, Febrero 02). *Rafa Arjonilla*. Retrieved from Backend: <https://rafarjonilla.com/que-es/backend/>
- Asociación Española de Domótica e Inmótica. (2019, Octubre 1). *CEDOM*. Retrieved from CEDOM: <http://www.cedom.es/sobre-domotica/que-es-domotica>
- Asociación Española de Domótica e Inmótica. (2020, Noviembre 15). *CEDOM*. Retrieved from CEDOM: <http://www.cedom.es/sobre-domotica/que-es-inmotica>
- Badii, C., Pierfrancesco, B., Difino, A., Nesi, P., Pantaleo, G., & Paolucci, M. (2019). *MicroServices Suite for Smart City Applications*. Florencia: MDPI.
- Bailey, J., Budgen, D., Turner, M., Kitchenham, B., Brereton, P., & Linkman, S. (2007). *Evidence relating to Object-Oriented software design: A survey*. Durham: IEEE.

- BeJob. (2017, Febrero 14). *QUÉ ES LA PROGRAMACIÓN CON ARDUINO Y PARA QUÉ SIRVE*. Retrieved from BeJob: <https://www.bejob.com/que-es-la-programacion-con-arduino-y-para-que-sirve/#:~:text=La%20plataforma%20Arduino%20se%20programa,un%20sistema%20operativo%20llamado%20UNIX>.
- Bierzynski, K. (2019). Supporting the Self-Learning of Systems at the Network Edge with Microservices. *International Conference and Exhibition on Integration Issues of Miniaturized Systems*, 1-8.
- Blancarte, O. (2014, Julio 23). *Oscar Blancarte Software Architect*. Retrieved from Oscar Blancarte Software Architect: <https://www.oscarblancarteblog.com/2014/07/23/que-es-service-oriented-architecture-soa/>
- Blancarte, O. (2018, Junio 8). *Oscar Blancarte Software Architect*. Retrieved from Reactive Programming: <https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/microservicios>
- CEAC. (2021, Febrero 14). Retrieved from CEAC: <https://www.ceac.es/blog/pros-y-contras-de-la-domotica>
- Chakray. (2020). *¿QUÉ DIFERENCIAS HAY ENTRE REST Y SOAP?* Sevilla: Edge of the Web.
- Chávez, A., & Herrera, E. (2020). *Diseño e Implementación de una Smart Meter de Energía Eléctrica enlazado en una plataforma de visualización para Monitoreo y Control del consumo energético domiciliario basado en IoT*. Sangolquí: ESPE.
- Chavez, S., & Herrera, E. (2020). *Diseño e Implementación de una Smart Meter de Energía Eléctrica enlazado en una plataforma de visualización para Monitoreo y*

Control del consumo energético domiciliario basado en IoT. Sangolquí:

Universidad de las Fuerzas Armadas, ESPE.

Chawla, A. (2021). Intelligent Monitoring of IoT Devices using Neural Networks.

Conference on Innovation in Clouds, Internet and Networks and Workshops

(ICIN), 137-139.

Chen, S., Xu, H., Liu, D., Hu, B., & Wang, H. (2014). A Vision of IoT: Applications,

Challenges and Opportunities with China Perspective. *IEEE INTERNET OF*

THINGS JOURNAL, 349-358.

Cobo Jaramillo, N. M. (2021). *Diseño de una arquitectura para el control telemático de*

sistemas ciber físicos basada en microservicios y con tolerancia a fallos.

Sangolquí: Universidad de las Fuerzas Armadas - ESPE.

Dewesoft. (2020, Marzo 13). *Dewesoft d.o.o*. Retrieved from Dewesoft:

<https://dewesoft.com/es/daq/que-es-un-sensor>

Dominguez, J. (2007). *El reto de los servicios Web*. Cadiz: FLOSS International.

eCityclíc. (2020, Abril 20). *eCityclíc*. Retrieved from eCityclíc:

<https://www.ecityclíc.com/es/noticias/que-es-soa-o-arquitectura-orientada-a-servicios>

Ferreira, M. (2014). CAMINO AL HOGAR INTELIGENTE. *Informes de la Construcción*,

6-10.

Gatling. (2013, 07 29). *Gatling*. Retrieved from Gatling: <https://gatling.io/open-source/>

Guinard, D., & Trfia, V. (2016). *Building the web of things*. New York: Manning

Publications Co.

Haque, E. (2019). IoT Based Home Automation System with Customizable GUI and Low

Cost Embedded System. *International Conference on Sustainable Technologies*

for Industry 4.0 (STI), 1-5.

- Herrera Quintero, L. F. (2015). Viviendas Inteligentes. *Ingeniería e Investigación*, 47-53.
- ITTGWEB. (2021, Febrero 26). *Ingeniería de Software*. Retrieved from Seguridad de Software: <https://ittgweb.wordpress.com/2016/05/29/4-1-seguridad-de-software/>
- Jain, A. (2019). Home Automation System using Internet of Things (IOT). *International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 300-305.
- Key Studio. (2021, Noviembre 12). *Key Studio*. Retrieved from Key Studio: https://wiki.keyestudio.com/KS0085_Keyestudio_Smart_Home_Kit_for_Arduino
- Kitchenham, B., & Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Wellington: ResearchGate.
- Kitchenham, B., Budgen, D., & Brereton, P. (2011). *Using mapping studies as the basis for further research – A participant-observer case study*. Staffordshire: ELSEVIER.
- Klotz, B. (2018). *A car as a Semantic Web Thing*. Bilbao: IEEE.
- Kunold, I. (2019). Semantic Interoperability in Cyber-Physical Systems. *International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 797-801.
- Laleh, T., Paquet, J., Mokhov, S., & Yan, Y. (2018). *Constraint verification failure recovery in web service composition*. Montreal: Elsevier.
- Li, L., & Chou, W. (2015). Designing Large Scale REST APIs Based on REST Chart. *IEEE International Conference on Web Service*, 631-638.
- Lyaskov, M. (2017). A practical implementation of smart home energy data storage and control application based on cloud services. *International Scientific Conference Electronics (ET)*, 1-4.

- Mahamud, S. (2019). Domicile - An IoT Based Smart Home Automation System. *International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, 493-497.
- Moguel Márquez, J. (2018). *Una arquitectura orientada a servicios y dirigida por eventos para el control inteligente de UAVs multipropósito*. Universidad de Extremadura, Tecnologías Informáticas. Badajoz: Universidad de Extremadura.
- Morales, C. (2010). *Estado del Arte: Servicios Web*. Bogotá: Universidad Nacional de Colombia,.
- Morales, M. J. (2021, Febrero 26). *Modusoperantic*. Retrieved from Descubrimiento de los Servicios: <https://www.modusoperantic.com/es/descubrimiento-de-los-servicios/>
- Muhammad, W., & Sadia, M. (2015). A Review on Internet of Things (IoT). *International*, 1-5.
- Mustafa, B. (2021). IOT Based Low-Cost Smart Home Automation System. *International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 1-6.
- Navarro, J. (2021, Febrero 3). *Luis Llamas*. Retrieved from Luis Llamas - Ingeniería, Informática y Diseño.: <https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/>
- Naylamp. (2021, Octubre 8). *Naylamp Mechatronics*. Retrieved from Naylamp Mechatronics: <https://naylampmechatronics.com/espressif-esp/153-nodemcu-v2-esp8266-wifi.html>
- Nestrategia. (2021, 02 25). *Nestrategia, tu agencia de posicionamiento web en Madrid*. Retrieved from NESTRATEGIA: <https://nestrategia.com/desarrollo-web-back-end-front-end/>

- NSF. (2020, Junio 1). *National Science Foundation*. Retrieved from NSF, Where Discoveries Begin: https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503286
- Nur Aziza, R., Siswipraptini, P., Jabbar, M. A., & Ruli Siregar, R. (2021). The IoT and Cloud Based Smart Home Automation for a Better Energy Efficiency. *IEEE-International Conference on ICT for Smart Society (ICISS)*, 1-6.
- Oracle. (2012). *Oracle9i Application Server Web Services Developer's Guide*. Austin: Oracle.
- Oracle. (2020, 02 24). *Java*. Retrieved from Java: https://www.java.com/es/about/whatis_java.jsp
- Paradigma Digital. (2018, Octubre 10). *Paradigma*. Retrieved from Paradigma: <https://www.paradigmadigital.com/dev/spring-cloud-consul-1-2-descubrimiento-microservicios/>
- Perry, S. (2017, Noviembre 5). *IBM*. Retrieved from IBM: <https://developer.ibm.com/es/languages/java/tutorials/j-spring-boot-basics-perry/>
- Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). *Systematic mapping studies in software engineering*. Viena: ACM-Digital Library.
- Rahman, I. (2019). Voice-activated Open-loop Control of Wireless Home Automation System for Multi-functional Devices. *International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, 1-4.
- Raju, L. (2021). Advanced Home Automation Using Raspberry Pi and Machine Learning. *7th International Conference on Electrical Energy Systems (ICEES)*, 600-605.
- Rosa Moncayo, J. M. (2018, Mayo 17). *OpenWebinars*. Retrieved from OW: OpenWebinars: <https://openwebinars.net/blog/que-es-rest-conoce-su-potencial/>
- Roy, R. M. (2021). Voice Controlled Home Automation System. *Seventh International conference on Bio Signals, Images, and Instrumentation (ICBSII)*, 1-6.

- Ruby. (2020, 02 24). *Ruby*. Retrieved from Ruby El Mejor amigo de un desarrollador:
<https://www.ruby-lang.org/es/about/>
- Sainel. (2019, Junio 26). *INEL S-A*. Retrieved from INEL S.A: <https://www.sainel.es/que-es-la-inmotica-y-cuales-son-sus-ventajas/>
- Salazar, E. (2017). *Diseño e implementación de una interfaz inteligente para la gestión de recursos en el hogar mediante Smart TV, a partir de modelos y servicios*. Sangolquí: ESPE.
- Salman, L., Salman, S., Jahangirian, S., & Abraham, M. (2016). «*Energy Efficient IoT-Based Smart Home*. Canonsburg: Ansys Inc.
- Sommaya, M., Ramaswamy, R., & Siddhart, T. (2015). Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications*, 1-9.
- SpringBoot. (2021, Marzo 4). *Spring*. Retrieved from Spring:
<https://spring.io/projects/spring-boot>
- The PHP Group. (2020, 02 24). *My PHP.net*. Retrieved from PHP:
<https://www.php.net/manual/es/intro-whatcando.php>
- Triboan, D. (2019). Fuzzy-Based Fine-Grained Human Activity Recognition within Smart Environments. *IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI)*, 94-101.
- Trunov, A. (2018). *Legacy Applications Model Integration to Support Scientific Experiment*. Moscú: IEEE.
- Universidad de Murcia. (2019, Febrero 11). *UM.es*. Retrieved from UM.es:
<https://www.um.es/docencia/barzana/DAWEB/2017-18/daweb-tema-1-introduccion-html-css.html>

- Universidad Internacional de Valencia. (2018, Marzo 21). *VIU*. Retrieved from Tres tipos de seguridad informática que debes conocer:
<https://www.universidadviu.com/int/actualidad/nuestros-expertos/tres-tipos-de-seguridad-informatica-que-debes-conocer#:~:text=La%20seguridad%20de%20software%20se,proporcionar%20integridad%20autenticaci%C3%B3n%20y%20disponibilidad.>
- UXpañol. (2017, 02 25). *UXpañol - Discusiones sobre Experiencia de Usuario*. Retrieved from UXpañol: <https://uxpanol.com/teoria/sistema-de-escalas-de-usabilidad-que-es-y-para-que-sirve/>
- Wang, W., Suparna, D., Zhou, Y., Huang, X., & Moessner, K. (2017). *Distributed sensor data computing in smart city applications*. Macau: IEEE.
- Wikipedia. (2020). *Transferencia de Estado Representacional*. Huntsville: Wikimedia Inc.
- Wikipedia. (2021, 02 21). *Wikimedia Inc*. Retrieved from Wikipedia la enciclopedia libre: <https://es.wikipedia.org/wiki/Python>
- Yue, J., Wu, X., & Xue, Y. (2020). *Microservice Aging and Rejuvenation*. Xi'an: World Conference on Computing and Communication Technologies.
- Zeng, D., Guo, S., & Cheng, Z. (2011). The web of things: A survey. *JOURNAL OF COMMUNICATIONS*, 50-62.
- Zhang, M. (2020). Intelligent business cloud service platform based on SpringBoot framework. *Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 201-207.
- Zheng, T., Zhang, Y., Zheng, X., Fu, M., & Liu, X. (2017). *BigVM: A Multi-Layer-Microservice-Based Platform for Deploying SaaS*. Melbourne: Fifth International Conference on Advanced Cloud and Big Data.

