ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

INTEGRACIÓN DE NUEVAS TECNOLOGÍAS A UN SISTEMA LIMS CLOUD

INTEGRACIÓN DE COMUNICACIÓN ENTRE DISPOSITIVOS IOT Y EL SISTEMA DE LABORATORIO LIMS

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SOFTWARE

BRYAN STEVEN PAUCAR VEGA

bryan.paucar01@epn.edu.ec

DIRECTOR: PhD. CARLOS EDUARDO ANCHUNDIA VALENCIA

carlos.anchundia@epn.edu.ec

DMQ, febrero 2024

CERTIFICACIONES

Yo, Bryan Steven Paucar Vega declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Bryan Steven Paucar Vega

Certifico que el presente trabajo de integración curricular fue desarrollado por Bryan Steven Paucar Vega, bajo mi supervisión.

Firmado electrónicamente por:
CARLOS EDUARDO
ANCHUNDIA VALENCIA

PhD. Carlos Eduardo Anchundia Valencia

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como los productos resultantes del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Bryan Steven Paucar Vega

Carlos Eduardo Anchundia Valencia

DEDICATORIA

Dedico este proyecto a mi madre Ana y a mi padre José para compartir el logro de obtener un hito más en mi vida académica, me apoyaron incondicionalmente en las dificultades que se presentaron a lo largo de mi vida académica con recursos, palabras de ánimo y consejos.

A mis hermanos para ser recíproco con el tiempo y las palabras de ánimo que me ofrecieron en mi vida académica, mostrándoles que todo esfuerzo tiene su recompensa y para logarlo es necesario sacrificarse y no dar el brazo a torcer ante las adversidades.

A mi novia quien con sus palabras y acciones me permitieron seguir con el desarrollo del trabajo, dándome palabras de ánimo y siendo mi confidente en las dudas y observaciones sobre las ideas de este proyecto.

A mis amigos porque me extendieron una mano de ayuda en los momentos más urgentes cuando realice este trabajo celebrando este logro como de ellos porque vienen por el mismo camino.

AGRADECIMIENTO

Agradezco a la empresa Sideralsoft y al laboratorio clínico Asistanet por permitirme realizar este trabajo de titulación con su ayuda y la de sus clientes, les agradezco por hacerme sentir parte de su familia y por permitirme conocer como es el ámbito laboral en el desarrollo de soluciones software para los laboratorios clínicos.

Agradezco a mi tutor de este trabajo Calos Anchundia por sus críticas y su guía para obtener un trabajo de calidad. Además, por permitirme realizar la tesis con su la empresa de la que forma parte dándome la oportunidad de realizar este trabajo pegado aún más a la realidad de hacer proyectos de software en el mundo laboral.

Agradezco a mi madre por estar siempre apoyándome con palabras de aliento y con los alimentos que preparaba siempre antes de ir a las clases sin importar el horario en que tenía que salir de mi casa. A mi padre por brindarme apoyo económico ya sea para desarrollar mis labores diarias o resolver dificultades en mi vida académica.

A mis hermanos por sus palabras de apoyo y perseverancia para lograr alcanzar un hito más en mi vida académica y que cuando ellos lo hagan estaré ahí para apoyarlos también.

A mi novia por acompañarme en este proceso siendo mi confidente y darme ánimos para continuar con el desarrollo del presente trabajo.

A mis amigos y compañeros que realice a lo largo de mi vida académica por darme una mano de ayuda en dificultades presentadas en las aulas o fuera de ellas.

ÍNDICE DE CONTENIDO

RESUMEN

En el primer capítulo se definen los objetivos que se desean alcanzar al finalizar el siguiente trabajo de integración, también se realiza una búsqueda sistemática de la literatura para definir el estado del arte sobre la integración de dispositivos IoT dentro de los laboratorios clínicos. Esta búsqueda sistemática permite obtener los estudios base para comprender los fundamentos teóricos, definir el alcance y desarrollar el presente trabajo de integración.

En el segundo capítulo se define la teoría que sustenta la integración del paradigma de internet de las cosas (IoT) en el ámbito de los laboratorios clínicos. En este apartado se especifica los conceptos de la red IoT, sus aplicaciones y la arquitectura que sigue. Además, se define a el desarrollo basado por especificaciones como metodología para el desarrollo del presente trabajo, detallando los entregables esperados por cada fase. Finalmente se define el concepto de API REST como una interfaz de comunicación para consultar y almacenar la información de la red IoT.

El tercer capítulo corresponde a el desarrollo simultáneo del API REST y la red de dispositivos IoT siguiendo la metodología definida en el capítulo dos. Se inicia con la creación de un diagrama de despliegue para visualizar la interacción que existe entre la red de dispositivos IoT y el API REST. Luego, se detalla el comportamiento del API por medio de una historia del API y su correspondiente diagrama de secuencia web. Además, se establece el diagrama de clases del middleware encargado de conectar el API REST con la red de dispositivos IoT. Posteriormente se simula el comportamiento del API para obtener *feedback*, que ayuda a su implementación. Para finalizar se llevan a cabo pruebas unitarias para constatar el funcionamiento individual de cada componente y las pruebas de integración para verificar la comunicación e intercambio de información entre la red de dispositivos IoT y el API REST.

Finalmente, en el capítulo cuatro se consolidan los logros obtenidos en base a los objetivos planteados y el desarrollo realizado, que se plasman en las conclusiones. Además, se brinda recomendaciones que faciliten el proceso de desarrollo del proyecto simplificando el esfuerzo requerido y sugiriendo formas de contrarrestar las falencias encontradas.

PALABRAS CLAVE: LoRaWAN, Internet of Things, API REST, Red IoT, LIMS, Smart Lab, Mqtt.

ABSTRACT

In the first chapter, the objectives to be achieved at the end of the next integration work are defined, and a systematic search of the literature is also carried out to define the state of the art on the integration of IoT devices within clinical laboratories. This systematic search allows us to obtain the basic studies to understand the theoretical foundations, define the scope and develop the present integration work.

The second chapter defines the theory that underpins the integration of the Internet of Things (IoT) paradigm in the field of clinical laboratories. This section specifies the concepts of the IoT network, its applications, and the architecture it follows. In addition, development based on specifications is defined as a methodology for the development of this work, detailing the expected deliverables for each phase. Finally, the concept of REST API is defined as a communication interface to query and store information from the IoT network.

The third chapter corresponds to the simultaneous development of the REST API and the network of IoT devices following the methodology defined in chapter two. It starts with the creation of a deployment diagram to visualize the interaction that exists between the network of IoT devices and the REST API. Then, the behavior of the API is detailed by means of a history of the API and its corresponding web sequence diagram. In addition, the class diagram of the middleware in charge of connecting the REST API to the network of IoT devices is established. Subsequently, the behavior of the API is simulated to obtain feedback, which helps its implementation. Finally, unit tests are carried out to verify the individual operation of each component and integration tests are carried out to verify the communication and exchange of information between the network of IoT devices and the REST API.

Finally, chapter four consolidates the achievements obtained based on the objectives set and the development carried out, which are reflected in the conclusions. In addition, recommendations are provided to facilitate the project development process by simplifying the effort required and suggesting ways to counteract the shortcomings found.

KEYWORDS: LoRaWAN, Internet of Things, API REST, network IoT, LIMS, Smart Lab, Mqtt.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

1.1 Introducción

Los laboratorios clínicos son instituciones de apoyo al área de la salud, encargados del análisis de muestras biológicas para detectar posibles enfermedades o afecciones presentes en los pacientes. La representación básica del proceso analítico que sigue un laboratorio clínico se separa en tres fases: preanalítica, analítica y post-analítica [1]. En la fase preanalítica se consideran las actividades de solicitud y obtención de la muestra, transporte, recepción y entrega para la siguiente fase. La fase analítica corresponde al análisis cuantitativo y cualitativo, el control de la calidad y la validación de las pruebas. Mientras que la fase post-analítica se realiza la elaboración del informe para su entrega, además, si se requiere el caso, la custodia y conservación de las muestras.

Pero este proceso analítico que siguen los laboratorios clínicos no está exento de errores tanto en sus faces principales como en las actividades que componen dichas fases [2]. Estos errores pueden ser cometidos por el personal involucrado en el análisis o por los procesos de aseguramiento de calidad en el manejo de las muestras. Los errores más frecuentes que se pueden surgir en la fase preanalítica se encuentran la solicitud incorrecta de exámenes, la falta de identificación del paciente, el incumplimiento de las condiciones de extracción de las muestras y errores en la conservación de las muestras, entre otras.

En la fase analítica, a pesar de la automatización en el manejo de las muestras, el aseguramiento del control de la calidad es una de las actividades más importantes que garantiza la obtención de resultados exactos y precisos [2]. Dentro de este control de la calidad se toma en cuenta la conservación de los reactivos clínicos bajo la temperatura y humedad indicadas por el fabricante a fin de evitar la pérdida de su efectividad al aplicarlos a las muestras clínicas.

Finalmente, en la fase post-analítica se puede encontrar errores con la transcripción errónea de los resultados, errores de comunicación de los resultados y el tiempo que tarda en comunicar dichos resultados con otro centro médico, personal médico o paciente.

Estos errores pueden ser contrarrestados con la integración de nuevas tecnologías que automaticen los procesos de análisis del laboratorio clínico y eviten la aparición de estos errores. Un paso significativo en esta dirección es la integración de un sistema de gestión de la información de un laboratorio denominado LIMS, que resuelve los problemas asociados a la gestión de la información de los pacientes y sus muestras [3]. Este es uno de los primeros pasos que se han sido insertados en los laboratorios clínicos al adoptar el

servicio Orión como un proveedor LIMS ecuatoriano, encargado de la gestión de la información de los pacientes y sus muestras en los laboratorios clínicos [4].

Otra tecnología que ha sido adoptada por los *Smart Labs* es la implementación de redes de dispositivos IoT capaces de automatizar el control de las tareas secundarias referentes al control de los parámetros ambientales dentro de un laboratorio. La automatización de este control permite a los laboratoristas dedicarse a las actividades importantes dentro del laboratorio y delegando actividades secundarias, en este caso el control de temperatura y ambiente, para evitar la pérdida de tiempo.

El siguiente trabajo propone la integración de tecnologías de internet de las cosas (IoT) a un LIMS para automatizar una de las actividades de monitoreo, como lo es el ambiente interno de los refrigeradores y congeladores destinados al almacenamiento los insumos clínicos. Esto conlleva la integración de una red de dispositivos IoT que consiste en los nodos finales, un *Gateway*, un servidor de red y un servidor de aplicación. Con este enfoque se permite establecer una infraestructura que se encarga de recolectar las mediciones de temperatura y humedad de los dispositivos de refrigeración de un laboratorio clínico. Lo que permitiría automatizar el monitoreo y control del estado ambiental interno de los refrigeradores y congeladores. Además, el LIMS requiere que acceder a la información de temperatura y humedad de los refrigeradores y congeladores de un laboratorio clínico para poder utilizarla. Por lo cual se desarrolla un API REST a fin de poner a disposición esta información para ser utilizada por el LIMS del laboratorio clínico.

1.2 Objetivo general

Optimizar los procesos de medición de la temperatura y humedad de los almacenes de los reactivos dentro de un laboratorio clínico mediante el uso de dispositivos IoT y su *Gateway* para recolectar datos y ponerlos a disposición del laboratorio.

1.3 Objetivos específicos

- Reducir el tiempo y esfuerzo requerido por los laboratoristas para monitorear los reactivos mediante el uso de dispositivos IoT conectados a la red con el fin de mejorar la eficiencia en el seguimiento del ambiente interno de los contenedores de los reactivos.
- Monitorear la temperatura y humedad de los almacenes de los reactivos clínicos mediante el uso de dispositivos IoT para recolectar la información en tiempo real sobre los eventos sucedidos dentro de los almacenes de los reactivos.

1.4 Alcance

El alcance de este componente se fundamenta en la idea de introducir nuevas tecnologías de la industria 4.0 dentro del funcionamiento de un laboratorio clínico para darle características de *Smart Lab*. El objetivo es la adopción de la tecnología IoT dedicadas al monitoreo de las condiciones internas de los refrigeradores y congeladores de un laboratorio clínico. Donde los laboratoristas almacenan los reactivos, muestras e instrumentos necesarios para el desarrollo de sus actividades y que deben conservarse bajo un umbral de temperatura y humedad definidos.

Debido a esta necesidad se da la creación de un prototipo de una red de dispositivos IoT dentro de un laboratorio clínico. Esta red implica la integración de nodos finales con sensores de temperatura y humedad, encargados de recolectar estas mediciones en los refrigeradores y congeladores presentes en un laboratorio. Dicha información es capturada por un *Gateway*, que actúa como centralizador de los eventos, para luego transferirlos a un servidor de aplicaciones que se encarga del manejo de la información. También, se tiene un servidor de red se encarga de la gestión de los dispositivos y *Gateway* dentro de la red de dispositivos IoT.

La información recopilada por la red de dispositivos IoT se comunica a el LIMS integrado en el laboratorio clínico a través de un API REST, esta comunicación se realiza por medio de peticiones para guardar la información de las mediciones u obtener información. Esto con el fin de tener persistencia en el registro de las mediciones y aumentando un servicio más al LIMS del laboratorio clínico.

1.5 Estado del arte sobre IoT aplicada en un *Smart Lab*

Un proceso de revisión sistemática de la literatura ayuda a la identificación de información base para comprender lo que conlleva aplicar la tecnología IoT en el entorno de los laboratorios clínicos. Este proceso se ve reflejado en la **Figura 1.1**, donde se observa las actividades que se realizar para obtener literatura base para el desarrollo del componente.

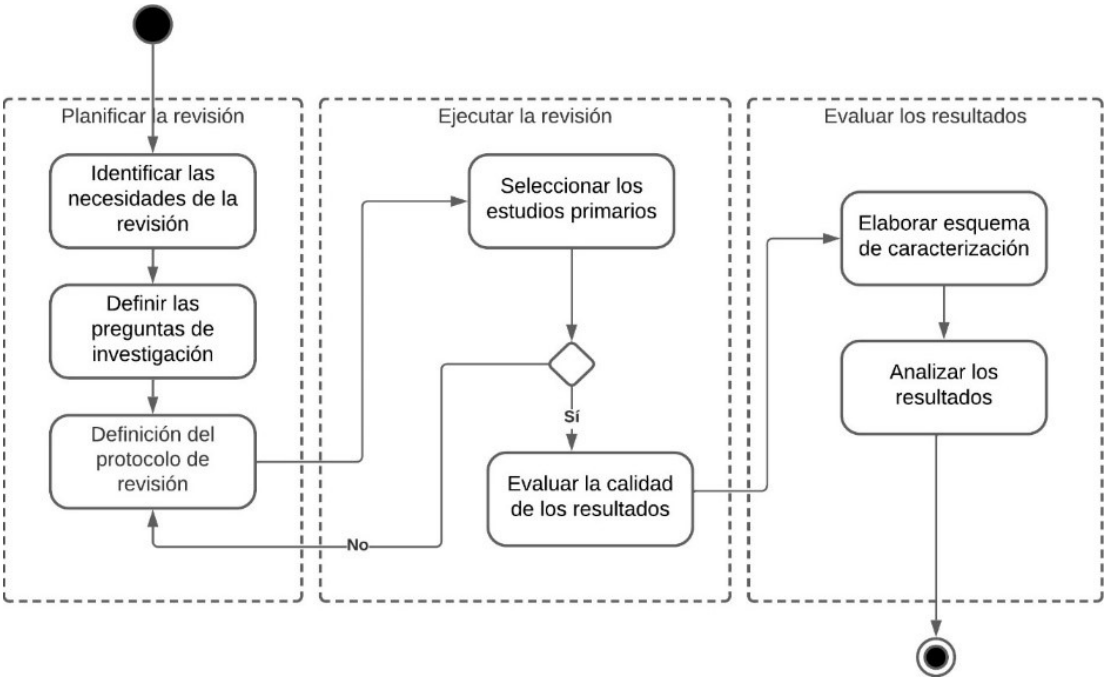


Figura 1.1. Proceso de revisión de la literatura. Adaptado de [5], [6], [7].

1.5.1 Necesidades de la revisión

Debido a que un *Smart Lab* integra múltiples tecnologías de automatización y que la tecnología IoT es una de ellas. Es necesario conocer el estado de los artículos actuales respecto a la integración de la tecnología IoT en la automatización de tareas dentro de los laboratorios clínicos. Para responder ciertas incógnitas sobre su aplicación en dicho entorno que se muestran a continuación:

1.5.2 Preguntas de investigación

Las preguntas que se buscan responder con el proceso de revisión de la literatura son las siguientes:

1. ¿En qué actividades y tareas se pueden integrar la tecnología IoT para automatizarlas dentro un laboratorio clínico, transformándolo en un *Smart Lab*?
2. ¿Qué recursos son necesarios para implementar la tecnología IoT dentro de un laboratorio clínico?
3. ¿Cómo ayuda la tecnología IoT en el desarrollo de las actividades de los laboratoristas?
4. ¿Cómo comunican los eventos o mediciones de las actividades realizadas por los dispositivos IoT a él laboratorio clínico?

1.5.3 Protocolo de revisión sistemática de la literatura

El protocolo de revisión permite establecer los criterios de inclusión, exclusión y de calidad, un marco temporal referencial, los términos de búsqueda, la cadena de búsqueda y las bases de datos de la búsqueda. Este protocolo permite filtrar la información hasta obtener la necesaria que responde las incógnitas planteadas.

Tabla 1.1. Determinación del protocolo para realizar la revisión sistemática de la literatura.[5]

Protocolo de revisión sistemática de la literatura	
Ámbito de la revisión	Tecnología IoT utilizados dentro de un laboratorio clínico.
Marco temporal	Fuentes desde el 2019 al 2023
Criterios de inclusión y exclusión	• IC-01: Se aceptan informes, tesis, documentación oficial de productos hardware o software, revistas y manuales de usuario. • IC-02: Estudios en español e inglés. • IC-03: Integración de la tecnología IoT en actividades dentro de un laboratorio clínico. • IC-04:Infraestructura de despliegue de la tecnología IoT dentro de un laboratorio clínico. • IE-01: Se excluye información de servicios pagados dados por empresas privadas. • IE-02: Estudios que tengan que ver con dispositivos IoT aplicados a la salud de los pacientes.
Criterios de calidad	• Especificación en la información. • Claridad de la información. • Orden de la información • Completitud de la información. • Precisión de la información
Fuentes de datos	*Research Gate*, Scopus, IEEE *Xplore, buscardo*.
Términos de búsqueda	*Components, network, clinical laboratory, API, monitoring, temperature, IoT, devices, communication, external systems, LIMS, integration.*
Cadena de búsqueda	*(“device IoT” OR “IoT technology” OR “IoT network” OR “sensors”) AND (“integration” OR “automation” OR “incorporation”) AND (“clinical laboratory” OR “LIMS” OR “laboratory”) AND (“monitoring” OR “control”) AND “internal environment”*

1.5.4 Seleccionar los estudios primarios

Aplicando el protocolo diseñado para la revisión sistemática de la literatura en las diferentes bases de datos, exceptuando los criterios de calidad, se obtiene las fuentes en las cuales se ahonda más allá de un vistazo rápido. El resumen de la bibliografía

analizada se presenta en el diagrama PRISMA[1] (véase en la **Figura 1.2**) para obtener los estudios que siguen el proceso de análisis de calidad.

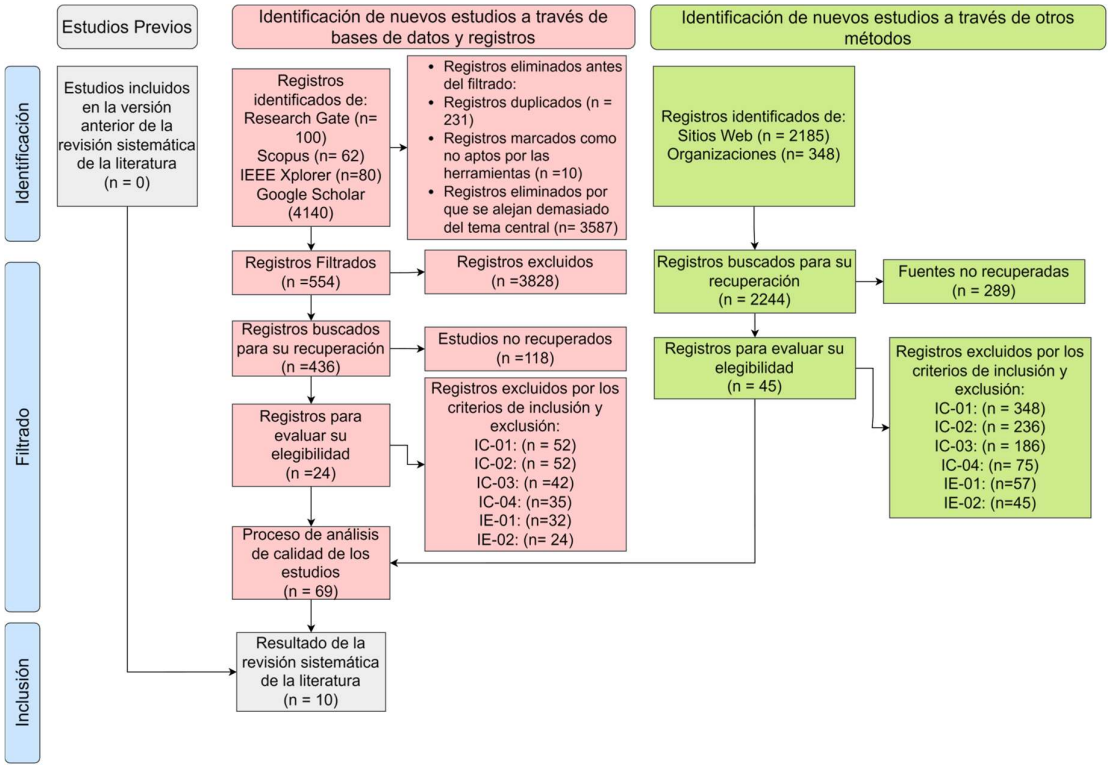


Figura 1.2. Diagrama de flujo PRISMA para el análisis del estado de la información técnica y científica.

1.5.5 Análisis de los resultados

Los criterios de calidad establecidos en la **Tabla 1.1** son aplicados a los artículos obtenidos en el diagrama de flujo PRISMA de la **Figura 1.2**. Este análisis consiste en una lectura profunda para escoger las referencias base para el desarrollo del presente componente, y descartar bibliografía con poca calidad de información. La literatura seleccionada se puede observar en la **Tabla 1.2**.

Tabla 1.2. Literatura base para el desarrollo del componente del proyecto.

Pregunta	Respuesta
¿ En qué actividades y tareas se pueden integrar la tecnología IoT para automatizarlas dentro un	En la industria 4.0 la tecnología de *Internet of things* (IoT) es una parte fundamental en la automatización de tareas. Otorgar a un dispositivo IoT las responsabilidades de monitoreo y control de los parámetros ambientales en un

[1] Diagrama de flujo PRISMA es utilizado para demostrar la aplicación de los criterios de inclusión, exclusión y de calidad para filtrar los artículos y trabajos encontrados en las bases de datos con la cadena de búsqueda. Esto permite evaluar y determinar los trabajos importantes referentes a el tema determinado.

laboratorio clínico, transformándolo en un *Smart Lab*?	espacio [8], [9]. Su implementación resulta en el aumento de la precisión y eficiencia de los experimentos [10]. Ya que los laboratoristas no se preocupan por actividades secundarias, centrándose en las principales.
¿Qué recursos son necesarios para implementar la tecnología IoT dentro de un laboratorio clínico?	Una arquitectura IoT por lo general consta de 6 capas: codificación, percepción, red, transporte, aplicación y negocio [11]. Cada una de estas capas cuenta con tecnologías aplicables que dan como resultado la arquitectura de red IoT. En particular, la tecnología *LoRaWAN* implementa una arquitectura de red tipo estrella que necesita tener los nodos finales *LoRaWAN,* un *Gateway LoRaWAN,* un servidor de red y uno de aplicación [12], [13]. Este es el hardware y software necesario insertar una red IoT dentro de un laboratorio clínico.
¿Cómo ayuda la tecnología IoT en el desarrollo de las actividades de los laboratoristas?	La implementación de una red IoT *LoRaWAN* permite transferir las responsabilidades de control y monitore de los parámetros ambientales en un laboratorio clínico [13], [14]. Dando seguimiento de los recursos primordiales de un laboratorio clínico.
¿Cómo se comunican los eventos o mediciones de las actividades realizadas por los dispositivos IoT a él laboratorio clínico?	La comunicación de una red IoT puede ser a través de diferentes tecnologías, entre estas esta LoRaWAN y WiFi. La tecnología LoRaWAN ofrece una comunicación de bajo consumo, a larga distancia y con un tamaño de la carga útil muy pequeño [8], [13]. En cambio, la tecnología WiFi transmite una carga útil más grande pero su consumo de energía aumenta y su alcance disminuye [15].

2 MARCO TEÓRICO

Esta sección comprende la base teórica para el desarrollo de este componente en base a los resultados obtenidos en la revisión sistemática de la literatura sección 1.5. Se detalla la tecnología aplicada a la arquitectura para la red IoT y su comunicación con sistemas externos por medio de un API REST.

2.1 *Internet Of Things* (IoT)

Internet de las cosas (IoT) es definido como una infraestructura global que propicia la interconexión de objetos físicos con las tecnologías de comunicación para la trasferencia de información [10], [11]. Estos dispositivos contienen partes de electrónica, software, sensores, actuadores y conectividad embebidos dentro del mismo; con el fin de comunicar la información con sistemas externos. Estos dispositivos son utilizados en varios contextos como la domótica, *e-health*, asistentes domésticos, aprendizaje mejorado, *Smart Lab*, etc.

2.1.1 Integración del IoT en un *Smart Lab*

Un *Smart Lab* es definido como un espacio diseñado para optimizar y automatizar los flujos de trabajo de un laboratorio proporcionando información en tiempo real de seguimiento y control de las condiciones, los equipos y los experimentos de un laboratorio [10]. Esto permite a los laboratoristas llevar a cabo sus tareas de manera precisa, eficiente y segura dando como resultado la reducción de costos y el esfuerzo que conlleva realizar tareas secundarias.

Internet of Things (IoT) es una de las tecnologías adoptadas por los *Smart Labs* cuyo potencial es aprovechado en el monitoreo del ambiente interno o externo al laboratorio. Permitiendo hacer un seguimiento continuo de medidas de temperatura, humedad, cantidad de luz, calidad del aire, entre otros [10]. Lo que minimizar el desperdicio de tiempo en la realización de esta tarea aumenta la productividad del laboratorio ya que se dedica más tiempo en el desarrollo de las tareas importantes y delegando las tareas secundarias a la automatización con dispositivos IoT.

2.2 Arquitecturas de *Internet Of Things* (IoT)

El paradigma IoT cubre muchos aspectos por lo que no existe una visión unificada en la que basarse para un modelo arquitectónico. Pero existen propuestas de arquitecturas en capas que se utilizan dentro en el desarrollo de redes de dispositivos IoT. Dichas arquitecturas pueden ser de tres, cuatro, cinco o seis capas con el fin de resolver el problema de la privacidad y la protección de datos del usuario [11].

Una propuesta arquitectónica de 6 capas (véase en la **Figura 2.1**) es mayormente utilizada en el desarrollo empresarial de soluciones IoT [11]. Este modelo parte desde la codificación del dispositivo para capturar información, el sensor que reúne la información, la transmisión de los datos, el procesamiento de los datos, la aplicación del dispositivo en un entorno y la generación de modelos de negocio. Todas estas capas poseen tecnologías y protocolos para pasar de una a otra.

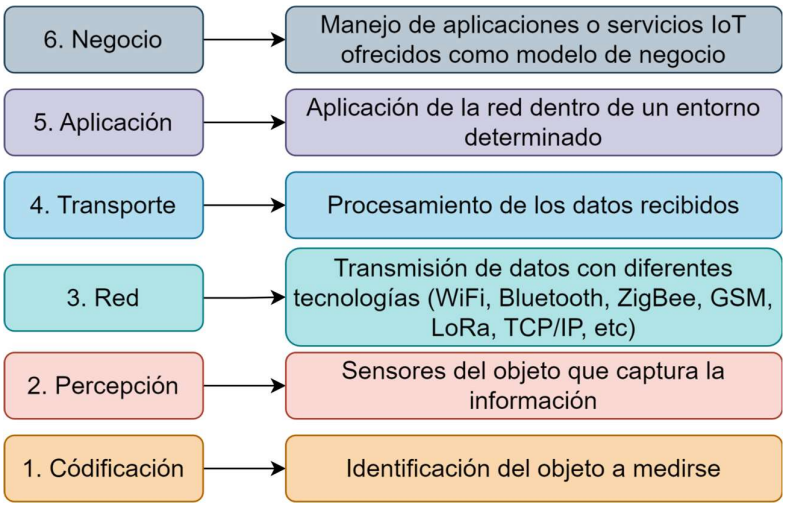


Figura 2.1. Modelo de arquitectura de 6 capas. Adaptado de [11]

Cada una de las capas de este modelo utiliza diferentes tecnologías y protocolos que permiten realizar si función y la interconexión con las demás capas de la arquitectura. La arquitectura toma forma en base a la arquitectura de red IoT que se va a construir.

2.3 Protocolos utilizados en la arquitectura IoT

Los protocolos utilizados en las diferentes capas permiten la interacción entre ellas y el manejo de la información desde los sensores hasta una aplicación. En la **Figura 2.2**, se puede observar los diferentes protocolos que pueden ser utilizados en conjunto o de forma individual en cada capa y como estos permiten la comunicación entre las capas.

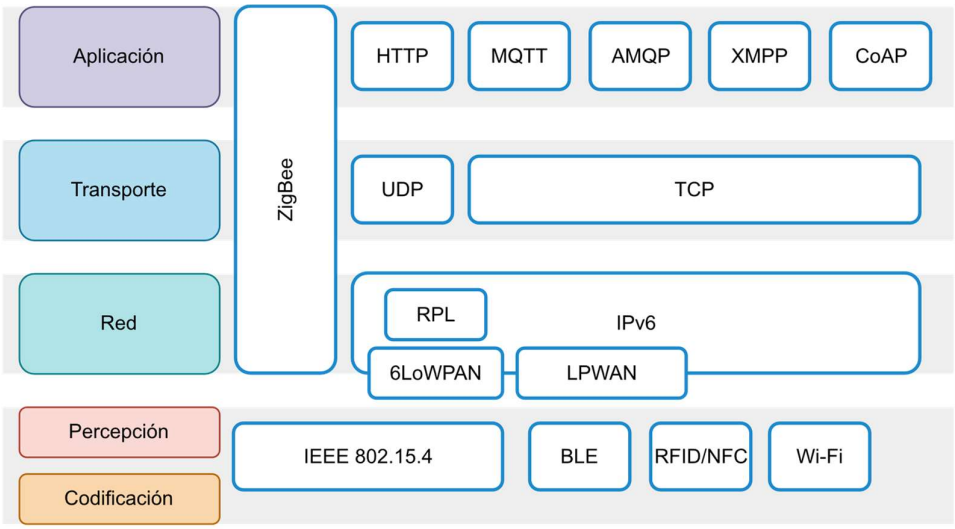


Figura 2.2. Protocolos utilizados en las capas del modelo arquitectónico. [11]

2.3.1 Protocolos de capa de aplicación

Los protocolos de capa de aplicación son orientados a la comunicación proceso a proceso de la información transmitida por los sensores. En la **Tabla 2.1** se puede observar las características principales de los protocolos de la capa de aplicación.

Tabla 2.1. Protocolos de la capa de aplicación. [11]

Protocolo	Características
MQTT	Denominado como *Message Queue Telemetry Transport* es un protocolo de comunicación ligero con poco consumo de recursos que funciona por medio del modelo suscriptor y publicador donde un publicador se encarga de enviar mensajes en una pasarela y el suscriptor escucha esos mensajes. Es ideal para la comunicar una red IoT a sistemas externos por su bajo consumo y forma de comunicación.
AMQP	*Advanced Message Queuing Protocol* es un protocol que utiliza varios modos de comunicación entre ellos están: la publicación y suscripción, el almacenamiento y reenvío, y el encolamiento de mensajes. Estos tipos de comunicación lo hacen un protocolo estable al entablar su canal de transferencia de datos entre un emisor y el receptor.
XMPP	*eXtensible Messaging and Presence Protocol* es un protocol utilizado en *streaming* por que permite una comunicación bidireccional entre el cliente, un *Gateway* y el servidor.
CoAP	*Constrained Applicaction Protocol* es un protocolo de mensajería basado en REST modificado de HTTP, pero con la carga de la comunicación más ligera. Tiene problemas como el retardo, el consumo y la transmisión de la información.

2.3.2 Protocolos de la capa de red

Los protocolos de la capa de red proveen la conectividad necesaria entre diferentes sistemas separados geográfica o tecnológicamente. En la **Tabla 2.2** se puede observar los protocolos mayormente utilizados en la comunicación.

Tabla 2.2. Protocolos de la capa de red. [11], [16]

Protocolo	Características
RPL	Es un protocolo de enrutamiento dedicado a redes de baja potencia y con pérdida de información. Crea de forma dinámica una topología con los nodos donde un nodo transmite la información a su padre hasta llegar a la raíz o el *Gateway*.
6LoWPAN	Este protocolo combina IPv6 con las redes LoPWAN lo que lo hace de bajo costo para ser conectados de forma inalámbrica. Es ideal para redes IoT compuestas de muchos dispositivos ya que transmite la información de forma comprimida bajando su coste de comunicación y su alta compatibilidad con sistemas heredados.
LPWAN	Es un protocolo de baja potencia, bajo consumo de energía y de largo alcance. Estas características lo hacen ideal para el uso de dispositivos IoT de forma inalámbrica por un periodo comprendido entre 5 a 10 años.

2.3.3 Tecnologías utilizadas en la arquitectura IoT

Las tecnologías utilizadas en la arquitectura IoT (véase en la **Figura 2.3**) utiliza los protocolos aplicados a las diferentes capas del modelo arquitectónico, estas tecnologías condensan las características de los protocolos junto con rasgos de seguridad y comunicación entre las diferentes capas de la arquitectura.

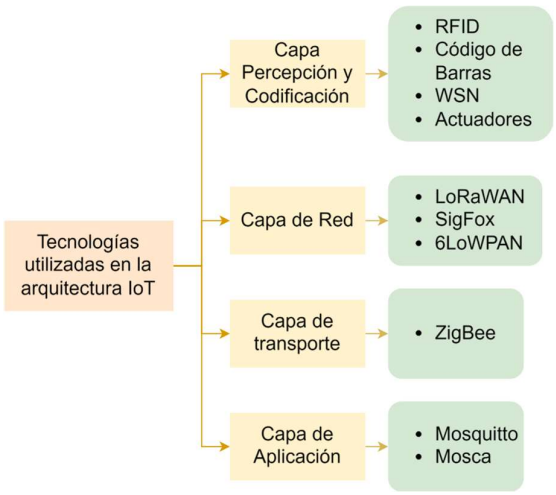


Figura 2.3. Tecnologías utilizadas en la arquitectura IoT. Adaptado de [11]

2.4 Arquitectura de comunicación en una red IoT

La arquitectura de comunicación en una red IoT muestra los componentes de la red con tareas definidas. Dentro de esta arquitectura de red se distribuyen las capas de la arquitectura IoT y los protocolos que implementa cada una de las capas.

Los componentes que integran una arquitectura de red IoT son los nodos finales, puertas de enlace (*Gateway*), servidor de red y servidor de aplicación como se puede ver en la **Figura 2.4**. En ciertos casos una tecnología puede desarrollar su propia arquitectura de red IoT, como es el caso de LoRaWAN que diseña una red bajo el protocolo LoRa como se aprecia en la sección 2.5.

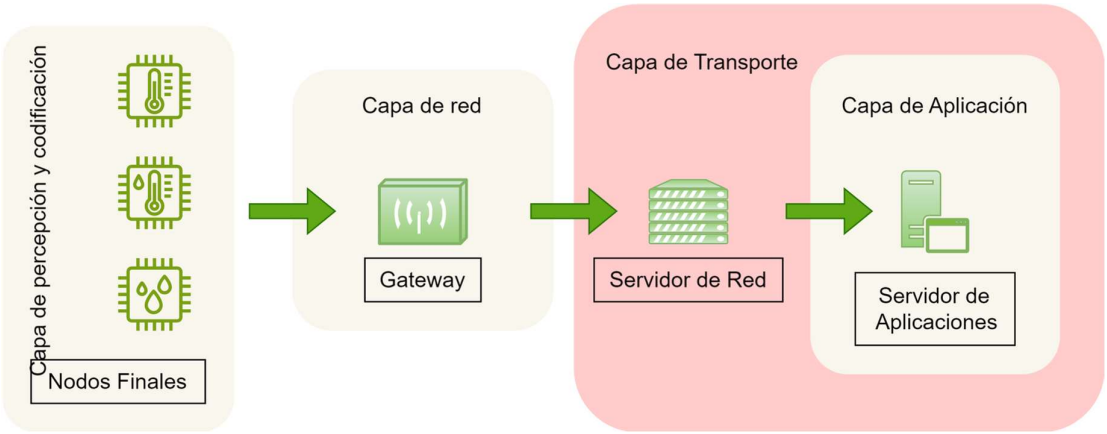


Figura 2.4. Arquitectura general de una red de dispositivos IoT. Adaptado de [16]

Las características de cada uno de estos elementos son las siguientes [16]:

- Nodos Finales: son los dispositivos IoT que tienen la programación para capturar un objeto mediante sus sensores.
- *Gateway*: Es el concentrador de la información de todos los nodos finales que se encuentran conectados a la red IoT.
- Servidor de Red: Es un servidor encargado de gestionar los nodos finales y *Gateways* presentes en la red IoT, además de agregar nuevos elementos a la red o eliminarlos.
- Servidor de Aplicaciones: Este servidor se encarga de aplicar la red de dispositivos IoT en un contexto determinado, se encarga de iniciar la red de dispositivos IoT en la comunicación de la información.

2.5 Arquitectura de red IoT LoRaWAN

La arquitectura de red LoRaWAN sigue una topología de tipo estrella como se puede ver en la **Figura 2.5** donde las puertas de enlace (*Gateway*) se encuentra en el centro de la red para trasmitir los mensajes hacia el servidor de aplicaciones [12]. Además, esta arquitectura tiene un servidor de red central que conecta las puertas de enlace por medio de conexiones IP estándar. La implementación *LoRa* permite la comunicación de los nodos finales a una o varias pasarelas sin requerir muchos recursos energéticos.

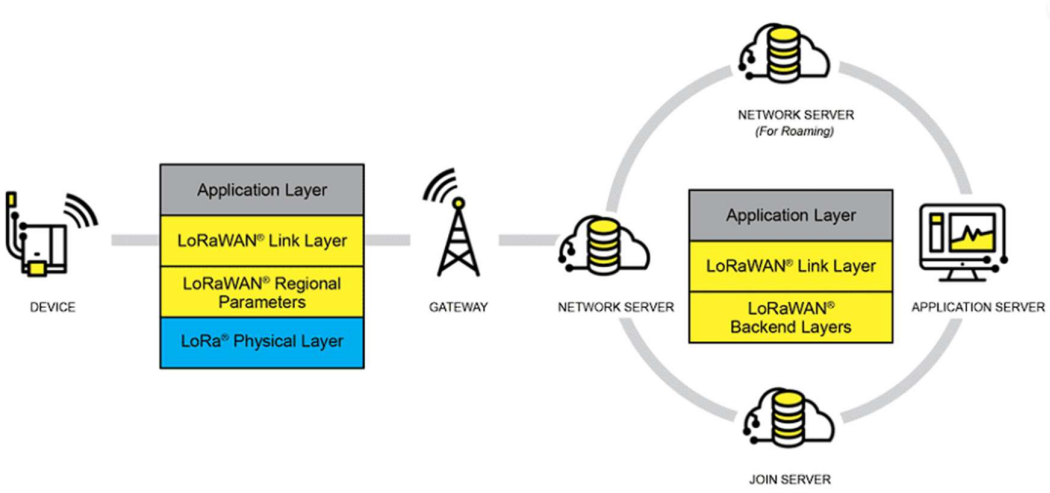


Figura 2.5. Arquitectura de red IoT basado en LoRaWAN. [12]

En esta arquitectura de red se aumenta un *Join Server* que se encarga de la autenticación de los nodos finales por parte del servidor de red. Este cambio permite determinar si al registrar un nuevo nodo, este cumpla con los parámetros necesarios y se verifique que es un nodo final.

2.5.1 LoRa y LoRaWAN

LoRa

LoRa es una implementación de la capa física del modelo OSI diseñada para la transmisión de datos a un largo alcance, pero con un coste de energía y recursos mínimos. Utiliza comunicación inalámbrica y la compresión de los datos capturados para lograr dichas características. La comunicación se realiza a través de radiofrecuencias desde los nodos finales a él *Gateway* correspondiente, dichas frecuencias son establecidas por LoRa Alliance para cada región del mundo (véase en la **Figura 2.6**) [12].

Región	Frecuencia (Hz)	Nombre
Europa	863-870 433	EU868 EU433
Norteamérica	902 - 928	US915
China	470 - 510 779 - 787	CN470 CN779
Autralia	915 - 928	AU915
India	865 - 867	IN865
Asia	923 - 1 923 - 2 923 - 3 923 - 4	AS923 - 1 AS923 - 2 AS923 - 3 AS923 - 4
Rusia	864	RU864

Figura 2.6. Frecuencias de LoRaWAN por regiones del mundo. Adaptado de [17], [18]

LoRaWAN

LoRaWAN es una especificación basada en el protocolo LPWAN en específico la implementación LoRa dirigido específicamente a la comunicación de los dispositivos IoT (nodos finales) con su *Gateway* [13]. Debido a su implantación LoRa también funciona bajo radiofrecuencias y permite una comunicación bidireccional entre el *Gateway* y los nodos finales dependiendo de su clase. La relación entre LoRa y LoRaWAN se puede observar en la **Figura 2.7**.

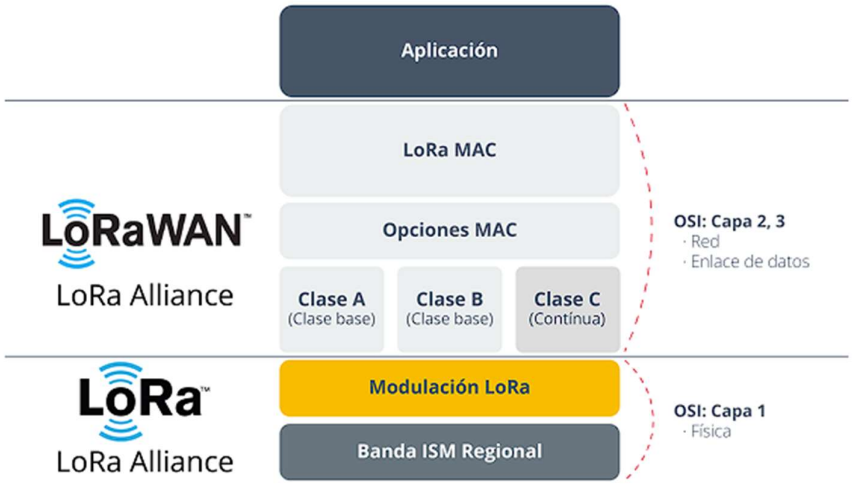


Figura 2.7. Relación entre LoRa y LoRaWAN según el modelo OSI. [13]

Clases de los nodos finales LoRaWAN

Como se puede observar en la **Figura 2.7** los nodos finales pueden tener tres clases (véase en la), cada una de estas clases tienen características que los hacen únicos y destinados a diferentes fines como se ve en la **Figura 2.8**. Las clases de los nodos finales se separan en [12]:

- Los nodos finales de clase A cuya característica principal es ser la clase predeterminada es el funcionamiento a menor potencia ya que envía la información en cualquier momento con un enlace ascendente y pasar a suspensión después de enviar los datos.
- Los nodos finales de clase B no envían información de forma esporádica sino mantiene una comunicación en un determinado horario permitiendo determinar una latencia de envío, esto hace que el consumo de energía sea mayor.
- Los últimos son nodos finales de clase C quienes bajan la latencia al mantener el canal de comunicación activo, pero se requiere energía continua para su funcionamiento.

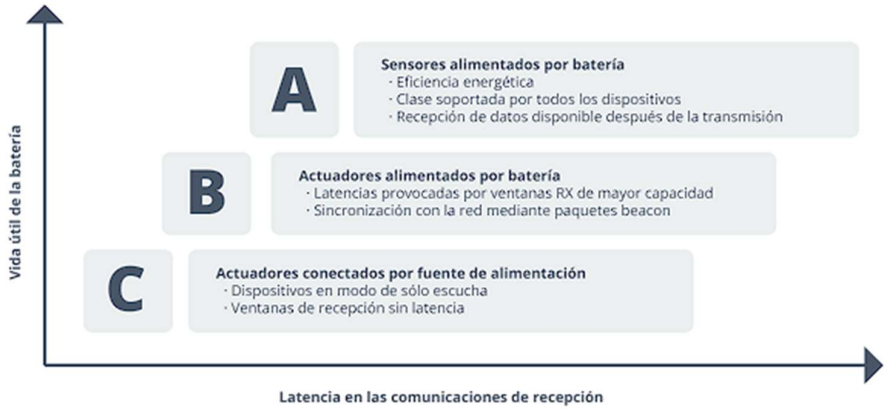


Figura 2.8. Clases de los nodos finales LoRaWAN. [13]

Formato de envío de información LoRaWAN

LoRA genera una trama (véase en la **Figura 2.9**) para el envío y recepción de los datos, esta trama comienza con un preámbulo que es un byte de sincronización y de diferenciación entre todos los nodos finales que están en una red LoRaWAN. Si este byte de sincronización no coincide con la configuración del *Gateway*, el nodo final no puede transmitir información. Después del preámbulo existe un encabezado. En este se indica el tamaño de la carga útil en bytes, la velocidad de transmisión y si hay o no CRC al final de

la trama. Lo siguiente en la trama es la carga útil que es la información que se envía y al final está el CRC que es la confirmación del tamaño de la carga útil.

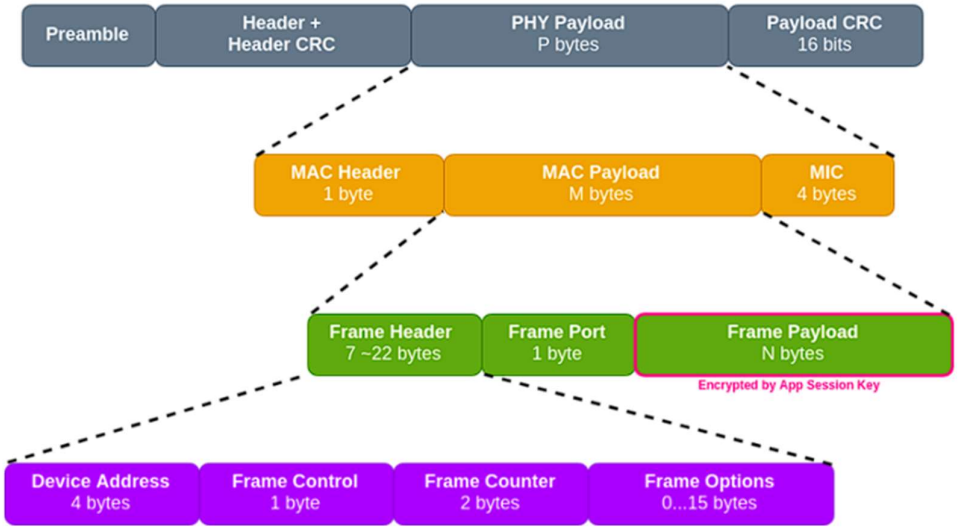


Figura 2.9. Formato de *frame* de LoRaWAN. Obtenido de [19]

2.6 API REST

Un API REST o API RESTful es una interfaz de programación ajustada a los límites arquitectónicos REST, lo que permite la interacción con servicios web. [20]

2.6.1 API

Un API es definida como el conjunto de protocolos y definiciones utilizados en la integración y diseño de software [20]. También son consideradas como una forma de compromiso entre el proveedor de la información y un consumidor, donde se establece las necesidades del consumidor y genera una respuesta del proveedor. Es decir, un API permite la interacción entre sistemas, computadoras y computadoras – sistemas; para generar una solicitud y que esta se cumpla.

2.6.2 REST

REST es un conjunto de límites de arquitectura y pueden implementarse de distintas maneras en la comunicación de los proveedores y consumidores. Cuando un cliente envía una solicitud, el proveedor responde por medio de HTTP la respuesta de en formato JSON, HTML, XLT, Python, PHP o un texto sin formato [20]. Las solicitudes pueden ser de tipo crear (*POST*), obtener (*GET*), modificar (PUT) y eliminar (*DELETE*) para cualquier *end point* disponible en el API.

Los encabezados incluyen la información de identificación y autorización para realizar alguna, ninguna o todas las solicitudes dependiendo del usuario. Esto junto a los códigos de respuesta son parte de un API RESTful bien diseñada.

2.7 Desarrollo Dirigido por Especificaciones (*Spec Driven-Development*)

El futuro de las aplicaciones es la creación de múltiples servicios como bloques o componentes cuya comunicación entre ellos o con sistemas externos se lo realiza a través de un API [21]. Esta forma de comunicación hace que la creación de un API tome más importancia, siendo necesario tomar en cuenta el ciclo de vida de un API (véase en la **Figura 2.10**) para su desarrollo [21]. Este ciclo de vida se centra en el producto que se desea alcanzar contemplando la especificación antes de la creación de un API.

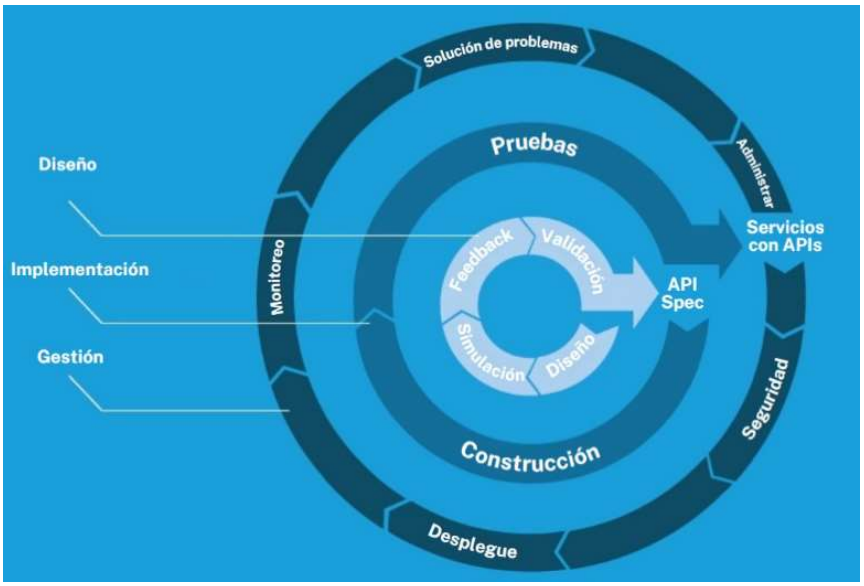


Figura 2.10. Ciclo de vida de un API. Adaptado de [21]

Como se observa en la **Figura 2.10**, cada una de estas fases contiene diferentes actividades complementadas con la cobertura ofrecida por Mike Amundsen en su libro "Design and Build Great Web APIs Rebust, Reliable and Resilient" para la concepción de un API robusta y escalable [22]. En dicho libro se destaca los entregables ofrecidos por cada fase del ciclo de vida del API como se puede observar en la **Tabla 2.3**.

Tabla 2.3. Entregables por fase del ciclo de vida del API

Fase	Entregables
Fase de diseño	• Diagrama de despliegue del componente (véase en la **Figura 3.1**) • Las historias del API (véase en la **Tabla 3.5**) • Los diagramas de secuencia web(véase en la **Figura 3.5**)
Fase de implementación	• Prototipo del API funcional (véase el Anexo V) • Red LoRaWAN (véase la sección 3.4.1) • Aplicación (Medicion.java) encargada de interconectar el API con la Red LoRaWAN (véase en Anexo V).
Fase de gestión	• Prototipo de API funcional con seguridad JWT (véase el Anexo V)

2.7.1 Primera Fase: Diseño

En la fase de diseño se reconoce los procesos que tiene que realizar el API sigue una perspectiva de afuera hacia adentro, es decir primero se decide como se ve y comporta el API antes de codificar su lógica. Esta fase comprende de tres actividades: el diseño preliminar, la simulación, *feedback* y la validación como se ve en la **Figura 2.11** [21]. Al realizar estas actividades dan como resultado la especificación del API.

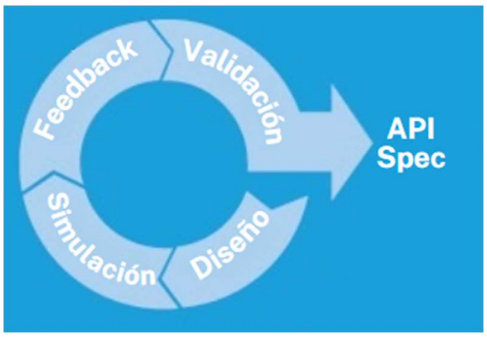


Figura 2.11. Fase de diseño del ciclo de vida de un API. Adaptado de [21]

Para la actividad de diseño Amundsen destaca dos pasos claves para determinar las necesidades y la concepción inicial del API, los cuales son el flujo de trabajo que seguirá el API y la definición de la historia del API [22]. Este flujo permite identificar los ciclos de trabajo donde se va a desenvolver el API que se plasma en un diagrama de secuencia web.

Mientras que la historia del API consiste en la creación de un documento explicativo donde se muestre que va a realizar el API y las acciones que animan esta historia [22]. Para escribir esta historia se sigue un formato simple con 5 secciones: el propósito que define la razón del API, los datos de entrada y salida, las acciones que se espera que realice el API y el procesamiento que es lo que se espera que haga el API.

La actividad de simulación consiste en convertir el diagrama de secuencia web en un *sketch* para transmitir las ideas de la actividad de diseño preliminar a un prototipo. Este prototipo funciona para ver el comportamiento del API utilizando Strapi. Dicho comportamiento se ve reflejado en definir escenarios correctos e incorrectos para ver el comportamiento y respuesta del API a los escenarios propuestos.

Una vez obtenido un prototipo de primera vista se presenta para obtener el *Feedback* correspondiente al comportamiento presentado del API. Al final con los cambios realizados sobre el prototipo y su posterior aceptación se valida el funcionamiento del API en base al cumplimiento de los requerimientos que se busca satisfacer.

2.7.2 Segunda Fase: Implementación

Esta fase consiste en pasar del prototipo inicial del API a un prototipo funcional mediante las actividades de construcción y pruebas como se ve en la **Figura 2.12**. La actividad de construcción se construye el API funcional utilizando el *framework* de *Express* de *NodeJS*. La siguiente actividad se prueba el API por medio de escenarios y su integración con la aplicación Medición

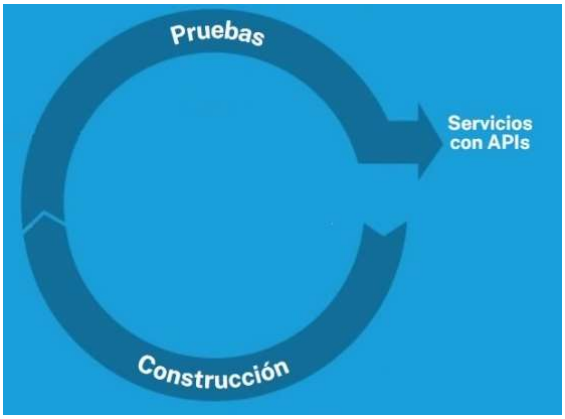


Figura 2.12. Fase de implementación del API

2.7.3 Tercera Fase: Gestión

En esta fase se hace llega hasta la actividad de seguridad en el API como se ve en la **Figura 2.13**. La actividad de seguridad consiste en implementar *Json Web token* (JWT)

para controlar el acceso de los usuarios a la información, este control permite que usuarios no autenticados o usuarios no pertenecientes a un rol no puedan obtener, enviar, modificar ni eliminar la información [22].

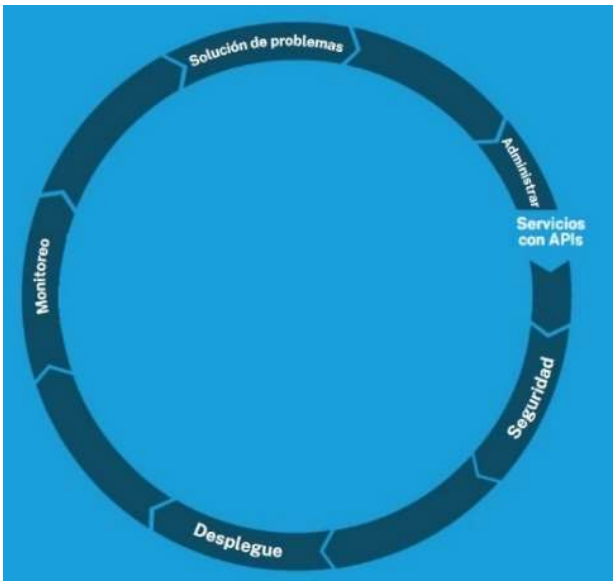


Figura 2.13. Fase de gestión del API.

2.8 Kick Off Meetings

Un *kick off meeting* es una oportunidad de reunir a al equipo de trabajo y los participantes del proyecto para compartir los detalles clave, así como la descripción del proyecto, sus objetivos y los próximos pasos que se darán [23]. De esta forma todos los involucrados en el desarrollo del proyecto dan sus aportes, lo que produce un equilibrio en el punto de vista sobre cómo abordar la problemática del proyecto. Además, el equipo esta alineado con la información necesario y con la mayoría de sus dudas resultas, pero, sobre todo, tanto los clientes como el equipo ya conocen lo que se espera del proyecto.

3 DESARROLLO

3.1 Preparación antes del proyecto

Antes de comenzar con el proyecto es necesario realizar diferentes *kick off meetings* para reunir a el equipo de trabajo con los *stakeholders*, buscando alcanzar un punto de inicio general, teniendo en cuenta el alcance y lo que se desea conseguir al finalizar el proyecto. En la **Tabla 3.2** se muestra el primer *kick off meeting* donde se define a las personas involucradas en el desarrollo del proyecto, junto con sus acciones y roles que toman en el desarrollo de este componente (véase en la **Tabla 3.1**).

Tabla 3.1. Identificación de las personas involucradas en el desarrollo del proyecto.

Persona	Rol	Actividades
Laboratorista	*Stakeholder*	Encargado del control y monitoreo de los insumos en los refrigeradores y congeladores del laboratorio clínico.
Arquitecto de soluciones de Sideralsoft	*Stakeholder*	Valida las fases del desarrollo conforme los requerimientos que se establecieron.
Tesista	Desarrollador	Encargado del desarrollo e inserción de la red LoraWAN de dispositivos IoT y del API REST.

Tabla 3.2. Detalles de los kick off meetings realizados antes de comenzar el proyecto.

Fecha	Lugar	Participantes	Temas tratados	Resultados
31/05/2023	Oficinas Sideralsoft	• Delegado de Sideralsoft • Desarrollador	• Conocer el ambiente de la empresa. • Presentación de la idea inicial de automatización del control del ambiente interno de los refrigeradores y congeladores del laboratorio con dispositivos IoT. • Establecer los interesados (laboratoristas y la empresa Sideralsoft).	• Aceptación de la idea de implementar una red IoT para el monitore y control del ambiente interno de un refrigerador o congelador. • Dar el servicio de monitoreo automático de los refrigeradores y congeladores de un laboratorio clínico.
18/09/2023	Oficinas Sideralsoft	• Delegado de Sideralsoft • Desarrollador	• Definir los recursos necesarios para desarrollar la red de dispositivos IoT junto con las tecnologías que se deben utilizar. • Definir la fecha de visita técnica a un laboratorio clínico.	• Se utiliza la tecnología de comunicación LoRaWAN dentro de la red IoT • La red estará conformada por dispositivos IoT con sensores de humedad y temperatura, además de un *Gateway*. • Asignación de una fecha para la visita al laboratorio clínico Asistanet.
21/09/2023	Asistanet laboratorio clínico	• Delegado de Sideralsoft • Desarrollador • Laboratorista	• Identificar los procesos internos del laboratorio clínico y donde se integra el componente de integración de dispositivos IoT. • Conocer como controlan y monitorean la temperatura y humedad de los refrigeradores y congeladore.	La red LoRaWAN IoT ayudara en la tarea de soporte al controlar la temperatura y humedad de los reactivos (véase en el ANEXO I).
23/09/2023	Oficinas Sideralsoft	• Delegado de Sideralsoft • Desarrollador	• Presentar las propuestas de dispositivos IoT y *Gateways* que cumplen las condiciones definidas. • Mostrar los requerimientos de datos que el API aceptará.	Dispositivos para utilizarse en la red son el sensor EM300TH y el *Gateway* UG 63, ambos productos de Milesight y la información que se obtiene (véase en el Anexo II).

3.2 Herramientas utilizadas

Nombre	Descripción	Utilizado	Logo
Milesight	Es un distribuidor de hardware y software destinado a el desarrollo de soluciones IoT.[24]	Red de dispositivos IoT (nodo final y *Gateway*), servidor de red, servidor Aplicación.	
Mosquitto	Agente servidor de mensajes enviados por el protocolo MQTT parte de la fundación Eclipse.[25]	Servidor MQTT	
Intellij idea	IDE de desarrollo integrado de aplicaciones de escritorio o web en Java.[26]	Aplicación escritorio	
Maven	Manejador de dependencias y de proyectos para Eclipse.[27]	Manejador de dependencias de la aplicación de escritorio	
XAMPP	Entre sus múltiples servicios se encuentra el servicio de base de datos SQL.[28]	Base de Datos	
Node JS	Es un entorno de ejecución en tiempo real de JavaScript.[29]	Aplicación web	
Swagger	Conjunto de herramientas para documentar el API.[30]	Documentación del API.	
Express JS	Es una infraestructura de aplicaciones web node JS.[31]	Estructura de la aplicación web.	

3.3 Primera Fase: Diseño

Esta fase empieza con la recolección de los requerimientos dados por los *stakeholders* para poderlos especificar en las historias de un API. Esta especificación permite diseñar la arquitectura de la solución y generar los diagramas de secuencia web que representen el proceso especificado en las historias del API.

3.3.1 Requerimientos

Tabla 3.3. Requerimientos del laboratorista.

Requerimientos	Laboratorista
Código	Requisito
L01	Registro automático de la temperatura y humedad de los refrigeradores y congeladores del laboratorio más de 3 veces al día indistintamente de la hora.
L02	Controlar la temperatura y humedad bajo los rangos establecidos para cada refrigerador y congelador, dependiendo de las especificaciones de los reactivos que se almacenan.
L03	Registrar el promedio, la media máxima y la medida mínima del día, tanto para la temperatura como para la humedad.
Observación: Estos requerimientos fueron obtenidos en base a la visita técnica a el laboratorio Asistanet especificado en el Anexo 1.	

Tabla 3.4. Requerimientos del Arquitecto de soluciones de Sideralsoft.

Requerimientos	Arquitecto de soluciones
Código	Requisito
A01	Crear una interfaz de programación de aplicaciones (API) que permita la consulta de las medidas de temperatura y humedad recolectadas en los congeladores y refrigeradores de los laboratorios clínicos.
A02	La información de todos los laboratorios debe darse por un solo acceso con un token para cada laboratorio.
A03	Crear una red IoT con utilizando el nodo final Milesight EM300-TH como sensor de temperatura y humedad, y el *Gateway* Milesight UG63 como concentrador de la información.
A04	Interconectar la red IoT con el API por medio de una aplicación de escritorio programado en java.
Observación: Revisar las formas de conectar el API con la red de dispositivos IoT para obtener la información de los eventos.	

3.3.2 Arquitectura de la solución del componente

El desarrollo del proyecto tiene dos aristas que deben desarrollarse en paralelo. La primera es el desarrollo de una red de dispositivos IoT dedicados al monitoreo de temperatura y humedad de los refrigeradores y congeladores del laboratorio clínico. Mientras que la segunda arista corresponde a el desarrollo del API REST para comunicar la información obtenida y guardada de la red de dispositivos IoT. La interacción de ambas aristas se puede observar en la **Figura 3.1**, donde se expone el diagrama de despliegue del proyecto.

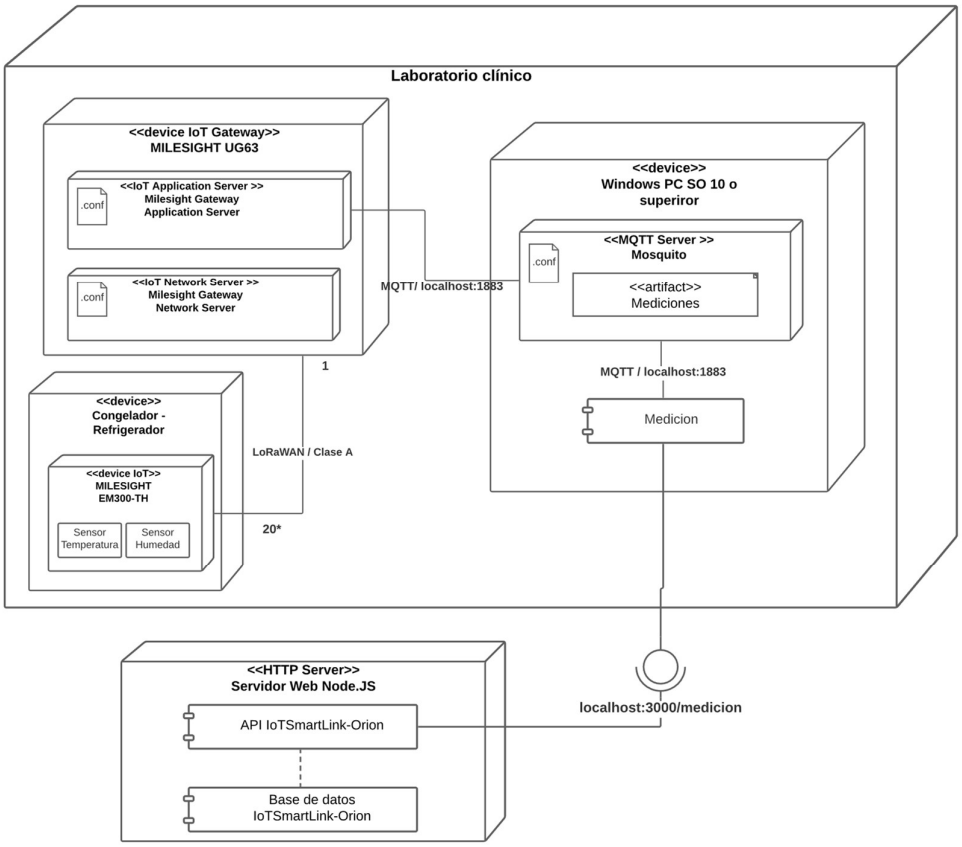


Figura 3.1. Diagrama de despliegue de la red LoRaWAN y su conexión con el API.

El diagrama de despliegue permite observa cual va a ser la interacción de los componentes de hardware y software. El hardware corresponde a la red IoT y el software a los componentes Medición, API IoTSmarkLink-Orion y la base de datos.

3.3.3 Componente Medición

Este componente es una aplicación que se suscribe al servidor Mosquitto para escuchar los mensajes que se publican desde el *Gateway* al servidor antes mencionado para trasmitir los datos de la medición. Además, esta aplicación de escritorio funciona de *Middleware* entre la red IoT y el API para enviar la información, convirtiéndolo en el cliente web para el API y se aplica su flujo en diagrama de secuencia que se ve en la **Figura 3.5**. la estructura de clases de la aplicación se puede observar en la **Figura 3.2**, donde se separa la lógica de la conexión con el servidor mqtt con la conexión con el API, pero realizando un tratamiento de datos intermedia.

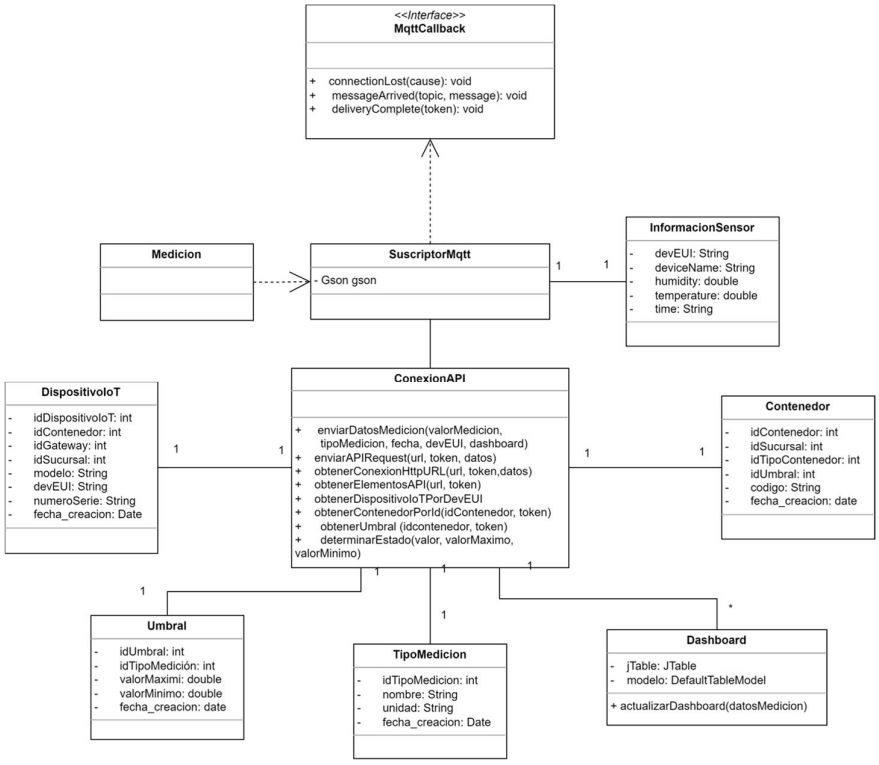


Figura 3.2. Diagrama de clases del componente Medición.

En la aplicación se puede observar que se establece una conexión de escucha como suscriptor con un servidor mqtt. Esta conexión permite la escucha de la información enviada por el *Gateway* LoRaWAN, y se ejecuta cada vez que existe una publicación nueva. Además, se establece una comunicación con el API y es por ello por lo que la aplicación Medición es un cliente web del API IoTSmartLink-Orion.

3.3.4 Componente base de datos API IoTSmartLink-Orion

El API necesita almacenar la información recolectada por la red IoT que se instala en un laboratorio clínico, la base de datos seleccionada es MySQL y su diagrama entidad relación sigue un enfoque multi-arrendatario como se ve en la **Figura 3.3**. Este enfoque permite tener una sola arquitectura de aplicación para diferentes clientes sin la necesidad de replicar el trabajo para cada uno de los clientes. Como se mostró en la **Figura 3.1**, el diagrama de despliegue se puede aplicar a los diferentes laboratorios, pero la comunicación es a una misma API con su base de datos, pero separando la información de los clientes dependiendo del laboratorio clínico.

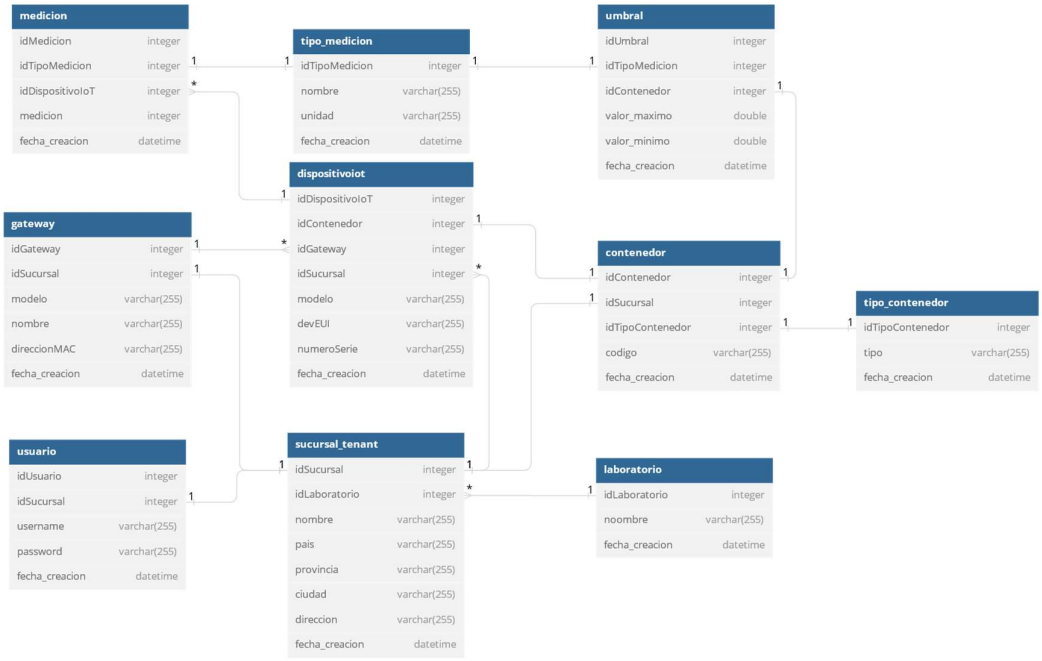


Figura 3.3. Diagrama entidad relación con el enfoque multi arrendatario para guardar las mediciones de la red IoT.

3.3.5 Componente API IoTSmartLink-Orion

Generación de la historia del API

La generación de la historia del API permite mapear los flujo y ciclos que debe seguir el API, estos procesos se materializan en la especificación de los requerimientos a través de una historia del API. Esta especificación ayuda a generar el diagrama de secuencia web que permite observar el flujo definido en la historia.

Mapeo del flujo general del API

Para automatizar le registro de las mediciones de temperatura y humedad es necesario que el componente Medición que se ve en la **Figura 3.1** debe seguir el proceso que se ve en la **Figura 3.4** para lograr guardar la información de las mediciones en el API por medio del *end point* para su registro.

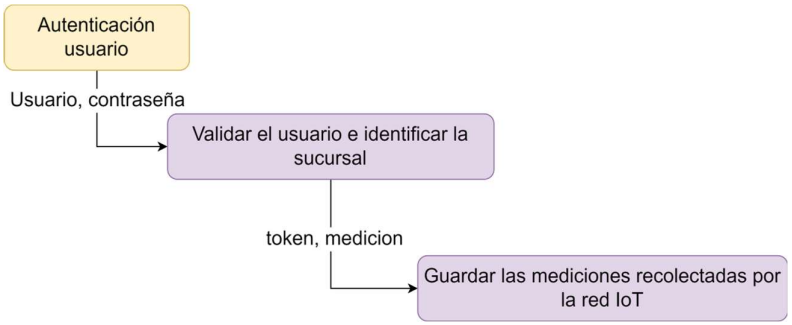


Figura 3.4. Flujo general del API.

Identificación de los ciclos internos del API

El mapeo de los flujos generales del API permite conceptualizar dos procedimientos, el primero es validar a un usuario con su usuario y contraseña. Mientras que el segundo es guardar la medición enviando una solicitud al API.

Un usuario registrado en el API puede generar un token de acceso, este token sirve para identificar y limitar las operaciones del API a una sucursal de un laboratorio clínico. El segundo proceso guarda una medición obtenida por el componente Medición en el API, para ello es necesario realizar procesos de verificación de los datos ingresados y se registra la medición dentro del API.

Historia del API

Para generar la Historia del API de los requerimientos obtenidos en las **Tabla 3.3** y **Tabla 3.4** respectivamente se consideran todos a excepción del requerimiento L03, esto debido a que el LIMS (Orión) cubre este requerimiento.

Tabla 3.5. Historia del API para el registro de una medición.

<table>
<tr><td colspan="2">Historia del API IoTSmartLink-Orion</td></tr>
<tr><td colspan="2">Propósito</td></tr>
<tr><td colspan="2">Automatizar registro y control de las mediciones de temperatura y humedad capturados por la red de dispositivos IoT permitiendo monitorizar y controlar estos aspectos de cada refrigerador o congelador de una sucursal de un laboratorio clínico.</td></tr>
<tr><td colspan="2">Datos</td></tr>
<tr><td>Entradas</td><td>Salidas</td></tr>
<tr><td>• Toke: token de acceso para una sucursal de un laboratorio clínico.
• Medición: objeto medición que se envía con la información: tipo de medición, valor de la medición, el estado y su fecha.</td><td>• Respuesta de medición registrada</td></tr>
<tr><td colspan="2">Acciones</td></tr>
<tr><td colspan="2">El API dependiendo del token accede a una sucursal y solo guarda la información de las mediciones asociada a dicha sucursal.</td></tr>
</table>

Reglas
• Debe existir un dispositivo IoT guardado en el API para asociarlo con la medición por su devEUI (identificador único de dispositivo IoT). • Debe existir un tipo de medición registrada para asociarla a la medición • Debe existir un contendor asociado al dispositivo IoT. • Debe existir un umbral asociado al contenedor para verificar el estado de la medición.
Procesamiento
• Validar el token • Validar los datos ingresados • Validar la sucursal • Crear la medición

Diagrama de secuencia web

Con el mapeo del flujo general del API y con la identificación de los ciclos internos del API, se puede obtener un esbozo del comportamiento del API que se plasma en un diagrama de secuencia web. En la **Figura 3.5** se observa el proceso para registrar una medición por parte del componente Medición que es considerado como el cliente web para el API.

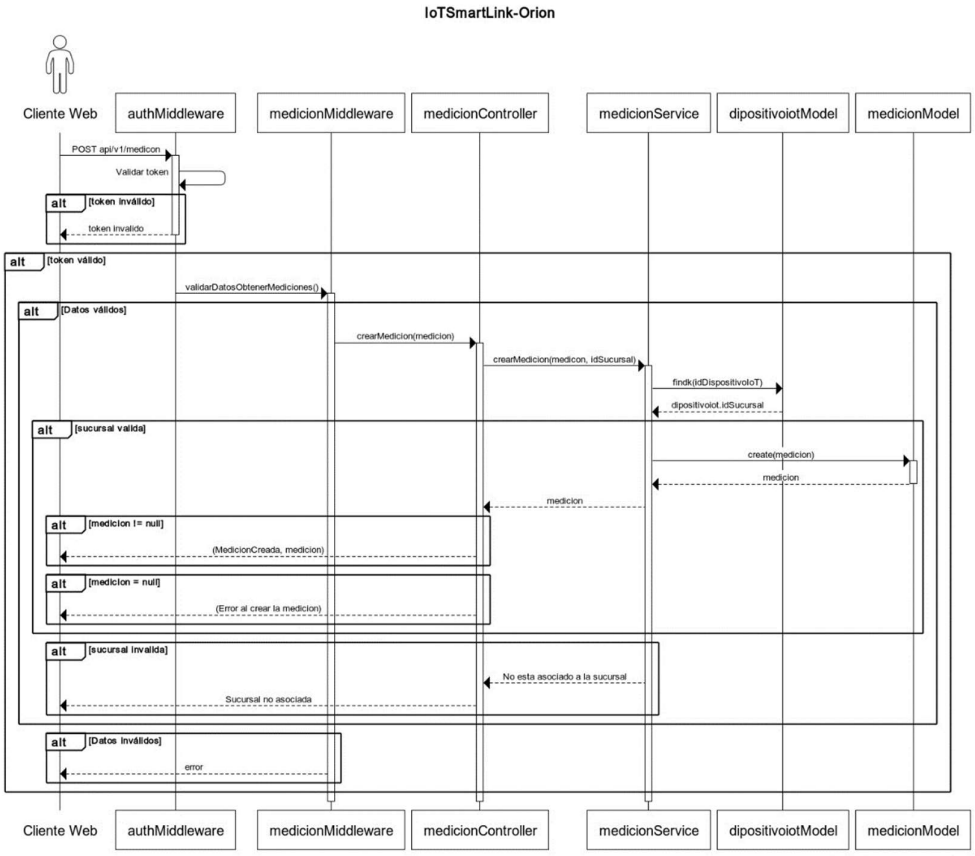


Figura 3.5. Diagrama de secuencia web para realizar el proceso de registrar una medición.

Creación del ALPS

El ALPS o *Application-Level Profile Semantic* es un documento de detalle del contexto general del flujo del API *IoTSmartLink-Orion* que se ve en el diagrama de secuencia en la **Figura 3.5**. En este documento se describen los elementos de entrada de cada uno de los flujos mostrados en dicho diagrama de secuencia y las operaciones que realiza el API con la información (véase en el Anexo III).

Simulación del comportamiento del API

Una simulación permite conocer cuál es el comportamiento del API *IoTSmartLink-Orion*, en base a el ALPS generado anteriormente y a los diagramas de secuencia web, ante posibles escenarios y observar los resultados que se obtienen. Es por ello por lo que se necesita crear ya un API con cierta funcionalidad, la cual es desarrollada con Strapi y generar escenarios para evaluar su funcionamiento fundamental.

El proceso para la creación del API *IoTSmartLink-Orion* simulada se puede observar en el Anexo III, donde ya se obtiene una cierta funcionalidad del API para poder realizar ciertas operaciones de camino feliz y camino triste. Utilizando la herramienta Postman se puede probar la funcionalidad del API determinando cual es el camino feliz y cual es un camino triste para la simulación.

Pruebas de la simulación

Escenario 1

En este escenario el usuario quiere obtener el token de acceso, pero su usuario o contraseña incorrectos:

Solicitud	POST
URL	http://localhost:3000/api/v1/login

Body	{
	"username": "Asistanet",
	"password": "Asistanet"
	}

Response	{
	"mensaje": "Credenciales inválidas."
	}

Escenario 2

El usuario ingresa datos de otro tipo como enteros, decimales o vacíos, pero no cadenas por lo cual el API le responde:

Solicitud	POST
URL	http://localhost:3000/api/v1/login

Body	{
	"username": 11232,
	"password": ""
	}

Response	{
	"errors": [
	{
	"type": "field",
	"value": 11232,
	"msg": "Formato incorrecto",
	"path": "username",
	"location": "body"
	},
	{
	"type": "field",
	"value": "",
	"msg": "El campo password es obligatorio.",
	"path": "password",
	"location": "body"
	}
	]
	}

Escenario 3

El ingreso de las credenciales del usuario es correcto lo que da como resultado obtener las credenciales y el token de acceso.

Solicitud	POST
URL	http://localhost:3000/api/v1/login

Body	{
	"username": "Asistanet",
	"password": "asistanet"
	}

Response	{
	"mensaje": "Usuario autenticado correctamente.",
	"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0I
	joxNzA3ODM3NDU2fQ.J-SKfBSFUmmF2gS77QlEqrwus7xMlNvS4kCVnEJP4HA",
	"usuario": {
	"idUsuario": 2,
	"idSucursal": 1,
	"username": "Asistanet",
	"password": "$2a$10$SXwo0t2r9Zrlk98MmQ8tdO19wWv
	Ca6s2voYqwMMKCozeTzF2NyJRO",
	"fecha_creacion": "2024-02-04T03:00:22.000Z"
	}
	}

Escenario 4

Se registra una medición con el token valido, los campos válidos y que la medición se asocie a una sucursal.

Solicitud	POST
URL	http://localhost:3000/api/v1/medicion
Autorización	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MiwiaWF0Ij oxNzA3ODM3OTIyfQ.zfMikUguGYTc7huGj45N9wD3wrLGtdWkvYZuHEhChg4

Body	{
	"idTipoMedicion": 3,
	"idDispositivoIoT": 1,
	"valorMedicion": 24.1,
	"estado": "Alerta",
	"fecha_creacion": "2024-02-11T12:22:56.000Z"
	}

Response	{
	"message": "Medición creada correctamente",
	"medicion": {
	"idMedicion": 423,
	"idTipoMedicion": 3,
	"idDispositivoIoT": 1,
	"valorMedicion": 24.1,
	"estado": "Alerta",
	"fecha_creacion": "2024-02-11T12:22:56.000Z"
	}
	}

Feedback del diseño del API

Tabla 3.6. *Feedback* obtenido en las presentaciones.

API	
Integración de un umbral	De forma automática si una medición va a ser ingresada desde el componente MedicionService se debe dar el estado dependiendo de los umbrales registrados en el API.
Integración de un tipo de medición	Las mediciones que no se guarde directamente, sino que se asocien a un tipo de medición para obtener el tipo de medición e incluso su unidad de medida (Ejemplo: tipoMedición: “Temperatura”, unidad: “Grados centígrados”)
Red LoRaWAN IoT	
Probar otro protocolo de comunicación para obtener la información	Es necesario probar otros protocolos de comunicación que el *Gateway* Milesight UG63 permite, en específico el http.
Implementar el componente Medición bajo un formato determinado por el arquitecto de soluciones	Utilizar un formato adecuado para generar la aplicación de escritorio Medición la cual se encarga de comunicar las mediciones al API.

3.4 Tercera Fase: Implementación

3.4.1 Implementación del componente medición

Este componente abarca la implementación de la red IoT LoRaWAN encargada de realizar las mediciones con sus sensores de temperatura y humedad y el desarrollo de la aplicación de escritorio para enviar dicha información al API. El primer procedimiento consiste en conectar el dispositivo IoT EM-300 TH con el *Gateway* UG63 por medio del protocolo LoRaWAN. Dado que los dos dispositivos son de la marca Milesight su procedimiento de conexión se puede realizar siguiendo el manual del Anexo IV y obtener la información como se ve en la **Figura 3.6**.

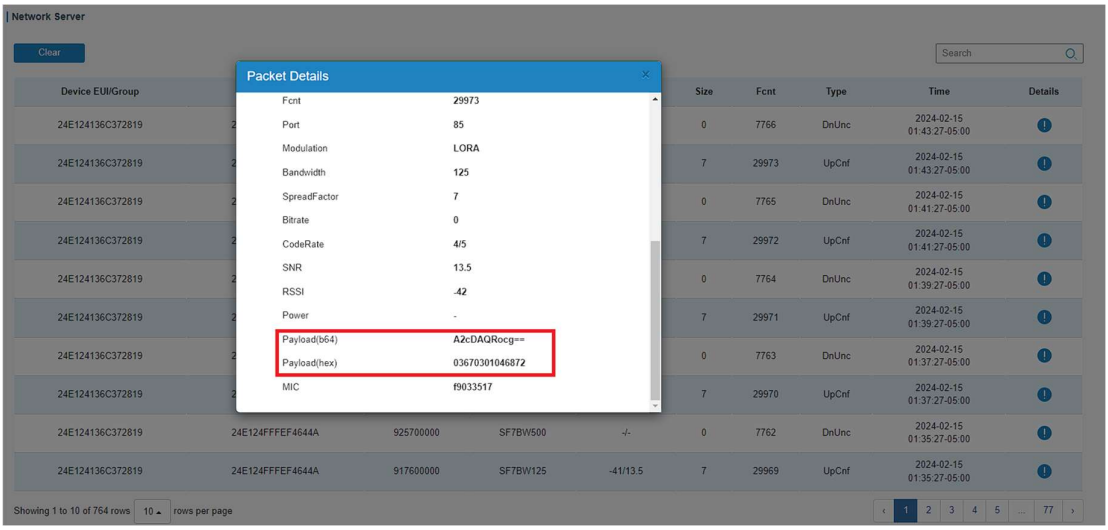


Figura 3.6. Envío de las mediciones de temperatura y humedad desde el dispositivo IoT al *Gateway* por medio de LoRaWAN.

Como se puede observar en la **Figura 3.6** la carga útil enviada por el dispositivo IoT al *Gateway* utiliza el formato especificado en la sección 2.5.1, este formato comprimido contiene la información de las mediciones de temperatura y humedad. El *Gateway* permite enviar esta información por pasarelas mqtt o http (revisar el Anexo V para instalar el bróker mqtt), específicamente se utiliza el protocolo mqtt para obtener una comunicación de baja potencia por medio del protocolo TCP/IP para las capas de transporte y de red como se ve en la **Figura 3.7**.

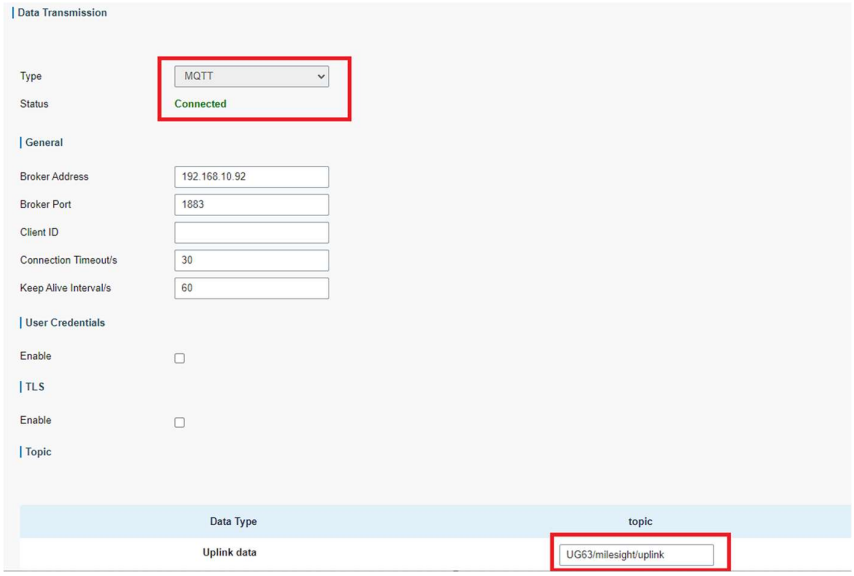


Figura 3.7. Conectar el *Gateway* con la pasarela mqtt a un canal de publicación.

Con esta configuración el *Gateway* está publicando las mediciones por el protocolo mqtt hacia un servidor Mosquitto. Con esta configuración en el componente Medición se puede implementar un suscriptor para escuchar los mensajes que se envía que se puede observar en el **Código 3.1** (véase en Anexo V para acceder al repositorio del componente Medición).

```
1. package com.connectionmqtt;
 2.
 3. import com.vista.Dashboard;
 4. import org.eclipse.paho.client.mqttv3.*;
 5.
 6. import com.envProperties.EnvProperties;
 7.
 8. /**
 9.  * Hello world!
10.  *
11.  */
12. public class Medicion
13. {
14.
15.         public static void main( String[] args )
16.     {
17.          Dashboard dashboard = new Dashboard();
18.          MqttSubscriber mqttSubscriber = new MqttSubscriber(dashboard);
19.          EnvProperties envProperties = new EnvProperties();
20.         String broker = envProperties.leerVariablesEntorno("BROKER_MQTT");
21.          String clientId = envProperties.leerVariablesEntorno("CLIENTID_MQTT");
22.          String topic = envProperties.leerVariablesEntorno("TOPIC_MQTT");
23.
24.
25.          try (MqttClient client = new MqttClient(broker, clientId)) {
26.              client.setCallback(mqttSubscriber);
27.
28.              MqttConnectOptions connOpts = new MqttConnectOptions();
29.              connOpts.setCleanSession(true);
30.
31.              System.out.println("Conectandose al broker: " + broker);
32.              client.connect(connOpts);
33.              System.out.println("Conectado");1.

34.
35.              System.out.println("Suscribiendose al topic: " + topic);
36.              client.subscribe(topic);
37.
38.          } catch (Exception   e) {
39.              System.out.println("Cliente Conectado" + e.getMessage());
40.
41.          }
```

```
42.     }
43.
44. }
```

Código 3.1. Código para conectarse a la pasarela del servidor con el protocolo mqtt.

En el **Código 3.2** en la línea 26 se da la escucha de los mensajes que se publiquen en el servidor mqtt, cada vez que exista una publicación se lanza el método para tratar los datos y enviarlos a un API.

```
 1. package com.connectionmqtt;
 2.
 3. import com.vista.Dashboard;
 4. import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
 5. import org.eclipse.paho.client.mqttv3.MqttCallback;
 6. import org.eclipse.paho.client.mqttv3.MqttMessage;
 7. import com.google.gson.Gson;
 8. import com.connectionapi.ConnectAPI;
 9.
10. public class MqttSubscriber implements MqttCallback{
11.
12.         private static Gson gson;
13.
14.
15.     public MqttSubscriber() {
16.                     MqttSubscriber.gson = new Gson();
17.     }
18.
19.         @Override
20.         public void connectionLost(Throwable cause) {
21.                     // TODO Auto-generated method stub
22.                     System.out.println("Connection lost because: " + cause);
23.         }
24.
25.         @Override
26.         public void messageArrived(String topic, MqttMessage message) {
27.                     // TODO Auto-generated method stub
28.                     String jsonPayload = new String(message.getPayload());
29.                     ConnectAPI connectAPI = new ConnectAPI();
30.
31.                      try {
32.                      // Convierte el JSON a la clase SensorData
33.                      SensorInformation sensorData = gson.fromJson(jsonPayload,
SensorInformation.class);
34.                                             System.out.println(jsonPayload);
35.
36.
37.                 } catch (Exception e) {
38.                     // Si no se puede convertir a JSON, simplemente imprime el payload como cadena
39.                     System.out.println(e);//arcivo en log
40.                                 //TODO: Implementar un log
41.
42.                 }
43.         }
44.         @Override
45.         public void deliveryComplete(IMqttDeliveryToken token) {
46.                     // TODO Auto-generated method stub
47.         }
48. }
```

Código 3.2. Suscriptor mqtt para escuchar las publicaciones.

3.4.2 Implementación del componente del API con la base de datos

La estructuración del desarrollo del API en base a los diagramas de secuencia web vistos en la **Figura 3.5**, se la hace por capas. Donde las capas son: modelos, servicios, controladores, rutas, *middlewares* y *helpers* como se puede ver en la **Figura 3.8** (véase el repositorio del código en el Anexo V)**.**

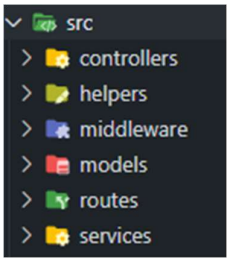


Figura 3.8. Estructuración del proyecto.

La carpeta de modelos muestra la estructura de la base de datos que se vio en la **Figura 3.3**, siguiendo el enfoque multi arrendatario la información se guarda asociándola a una sucursal del laboratorio clínico. El modelo para la entidad medición se puede ver en la **Figura 3.9**.

```
import DataTypes from 'sequelize';
import { database } from '../../database.js';
import { dispositivoIoTModel } from './dispositivoIoTModel.js';
import { tipoMedicionModel } from './tipoMedicionModel.js';

export const medicionModel = database.define('medicion', {
    idMedicion : {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true
    },
    idTipoMedicion: {
        type: DataTypes.INTEGER,
        allowNull: false
    },
    idDispositivoIoT: {
        type: DataTypes.INTEGER,
        allowNull: false
    },
    valorMedicion: {
        type: DataTypes.FLOAT,
        allowNull: false
    },
    estado: {
        type: DataTypes.STRING,
        allowNull: false
    },
    fecha_creacion: {
        type: DataTypes.DATE,
        allowNull: false
    }
},{
    tableName: 'medicion',
    timestamps: false
});

medicionModel.belongsTo(dispositivoIoTModel, { foreignKey: 'idDispositivoIoT' });
medicionModel.belongsTo(tipoMedicionModel, { foreignKey: 'idTipoMedicion' });
```

Figura 3.9. Modelo para la entidad medición que se guarda en la base de datos.

Por otro lado, en la carpeta rutas se tienen las rutas habilitadas para realizar las operaciones dentro del API, las rutas asociadas a las mediciones se muestran en la **Figura 3.10**. En la ruta se nota la aparición del *Middleware* para las mediciones antes de ejecutarse los métodos y asegurarse de que es un usuario habilitado y validar los datos de ingreso, para finalmente aplicar la operación desde el controlador hasta el servicio.

```
import { Router } from "express";
import { authMiddleware } from "../middleware/authMiddleware.js";
import { medicionController } from "../controllers/medicionController.js";
import { validarDatosObtenerMediciones, validarDatosIngresarMedicion } from "../middleware/medicionMiddleware.js";

const router = Router();

router.get("/medicion", authMiddleware, validarDatosObtenerMediciones , medicionController.obtenerTodasMediciones)
router.post("/medicion", authMiddleware,validarDatosIngresarMedicion, medicionController.crearMedicion)

export default router;
```

Figura 3.10. Rutas asociadas a las operaciones de las mediciones.

3.4.3 Pruebas Unitarias

Pruebas del componente del API IoTSmart-Link Orion

Las pruebas del API se realizan utilizando la herramienta de *Postman* para comprobar el código de la respuesta, el tiempo que se demora en obtener la respuesta y si los campos que se obtienen son los correctos. *Postman* permite programar estas pruebas para conocer su resultado como se ve en la **Figura 3.11** y se los resultados se muestran como pruebas pasadas, fallidas y omitidas.

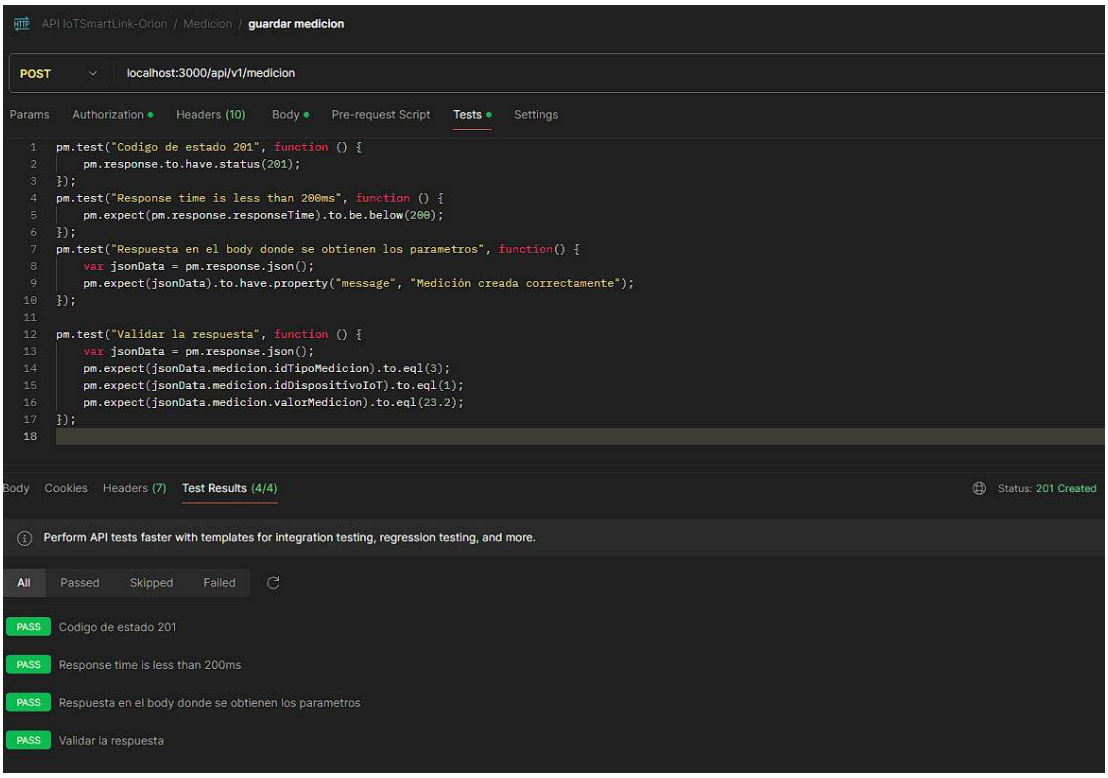


Figura 3.11. Pruebas para el *end point* de mediciones.

Tener programado las pruebas en *Postman* permite automatizarlas utilizando la herramienta de *Newman*, la cual permite obtener un resultado general para todas las

pruebas de los *end points* como se ve en la **Figura 3.12**. Pero también permite observar a detalle las respuestas obtenidas por cada prueba con la solicitud realizada y la respuesta obtenida, esta información se encuentra disponible en el Anexo V.

Figura 3.12. Resultados de las pruebas automatizadas en Newman.

Pruebas del componente Medición

Para probar que el *Gateway* Milesight UG63 está publicando los mensajes en el servidor Mosquitto utilizando un cliente mqtt como mqtt *explorer* para observar los datos publicados por el *Gateway* como se ve en la **Figura 3.13**.

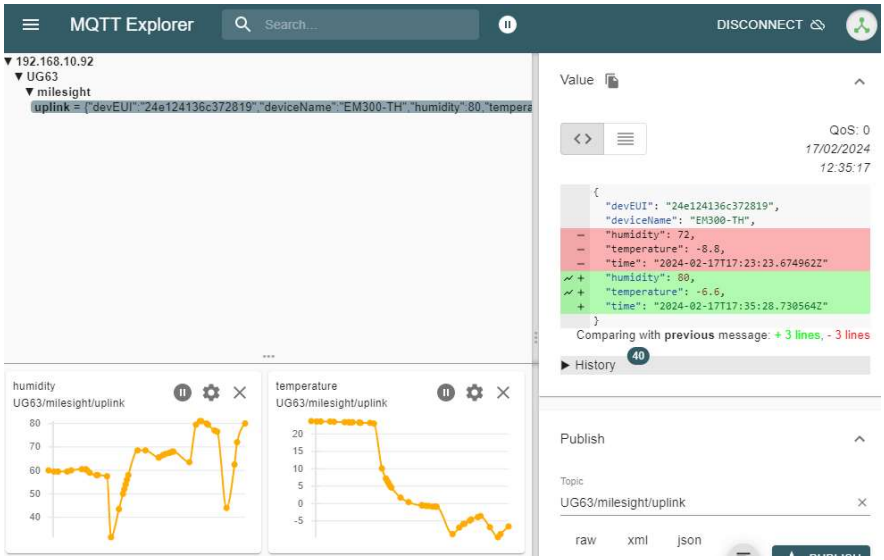


Figura 3.13. Datos publicados por el *Gateway* UG63 en el servidor Mosquitto.

Las publicaciones que realiza el *Gateway* en el servidor Mosquitto son escuchadas por el componente de Medición que está programado para escuchar un canal del servidor y se activa la llegada de los mensajes cada vez que exista una publicación como se ve en la **Figura 3.14**.


```
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":65.5,"temperature":-0.5,"time":"2024-02-17T15:25:23.643709Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":65.5,"temperature":-0.5,"time":"2024-02-17T15:25:27.051815Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":66.5,"temperature":-0.7,"time":"2024-02-17T15:31:23.643538Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":67,"temperature":-0.7,"time":"2024-02-17T15:35:23.639682Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":67.5,"temperature":-0.9,"time":"2024-02-17T15:41:26.873386Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":68,"temperature":-0.9,"time":"2024-02-17T15:45:27.912301Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":68,"temperature":-0.9,"time":"2024-02-17T15:47:23.645937Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":63.5,"temperature":-8.8,"time":"2024-02-17T16:11:23.660781Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":79.5,"temperature":-6.9,"time":"2024-02-17T16:21:23.652547Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":81,"temperature":-5.9,"time":"2024-02-17T16:27:23.662031Z"}
{"devEUI":"24e124136c372819","deviceName":"EM300-TH","humidity":81,"temperature":-5.8,"time":"2024-02-17T16:29:26.864457Z"}
```


Figura 3.14. Escuchar los mensajes publicados en un canal del servidor mqtt.

3.4.4 Pruebas de Integración

Este es el punto donde tanto el componente del API como el componente Medición se conecten para trasmitir la información obtenida de la red IoT hacia el API para guardarla. La integración se realiza en el componente medición que funciona de *Middleware* para tratar la información y enviarla por una petición *POST* al API. En la **Tabla 3.7** se define las pruebas de integración propuestas para conocer el resultado esperado cuando se desea alcanzar un objetivo en particular.

Tabla 3.7. Pruebas de integración de los componentes Medición y el API.

No	Objetivo	Descripción de la Prueba	Resultados Esperados
PI-01	Identificar si el dispositivo IoT está registrado en el API para guardar la medición.	Dado que una medición está asociada a un dispositivo IoT y un dispositivo IoT este asociado a un contendor permite conocer que la medición tiene fundamento para registrarla.	Si el dispositivo IoT se encuentra registrado en la base de datos devuelve la información de ese dispositivo IoT para ver a que contenedor está asociado.
PI-02	Identificar el umbral para determinar el estado de la medición si es normal o es una alerta.	Para automatizar el monitoreo de los contendores, es necesario que cada contenedor tenga un umbral aceptable de temperatura y humedad. Este umbral permite darle un estado normal o de emergencia al registro de la medición.	Se obtiene el umbral de la temperatura y la medición asociada a un contenedor que sirve para determinar el estado de la medición.
PI-03	Enviar una solicitud *POST* con la estructura de la medición para su registro	Una vez comprobada la información del dispositivo IoT, contendor y el umbral. Se crea el cuerpo de la solicitud donde se envía el idTipoMedicion, el idDispositivoIoT, el valor de la medición, el estado y la fecha.	Se obtiene una respuesta del servidor con la operación de creación de la medición es exitosa y la información de la medición creada.
PI-04	Mostrar la medición registrada junto con su contenedor y umbral asociado en la aplicación de escritorio.	En la aplicación de escritorio se muestra la información sobre la medición, el umbral y el contendor para seguir el control de la temperatura y humedad.	Se observa los registros realizados para el monitoreo y control de la temperatura y humedad.

3.4.5 Resultados

El punto de partida para comenzar el proceso de las pruebas de integración es la recepción de un mensaje que se publicó en una pasarela mqtt en el servidor Mosquitto, para proceder a realizar la consulta al API de la existencia del dispositivo IoT que capturo las mediciones por su *devEUI* como se observa en la **Figura 3.15** para cumplir con la prueba de integración PI-01.

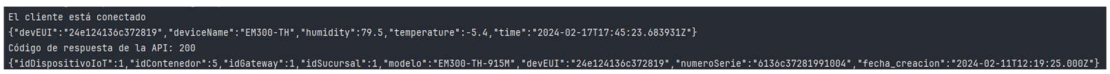


Figura 3.15. Resultado de la consulta del dispositivo IoT al API.

Una vez obtenida la información del dispositivo IoT se realiza la consulta para conocer la información de contendor asociado a el dispositivo como se ve en la **Figura 3.16** lo que cumple con la prueba de integración PI-02 y PI-03 por que se obtiene el umbral definido para el contendor.

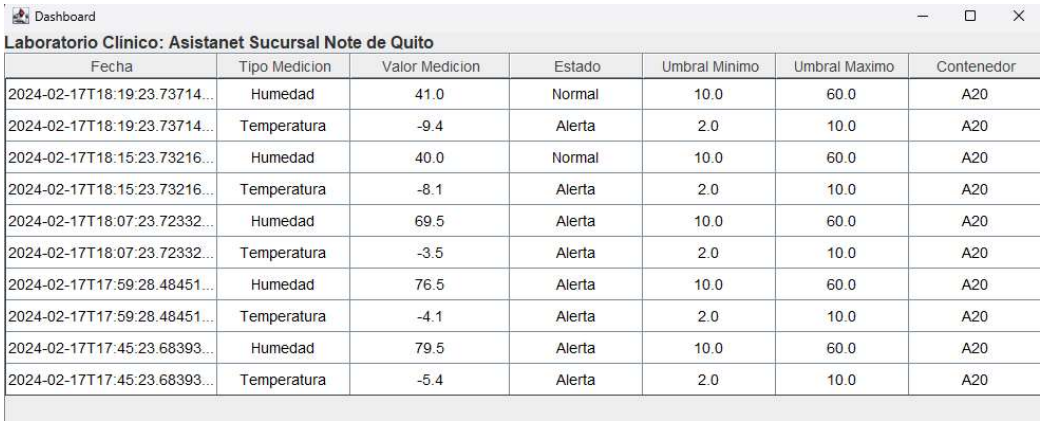
Figura 3.16. Obtener la información acerca del contendor.

Estas operaciones se aplican para ambos tipos de medición: la temperatura y la humedad por separados para poder armar el cuerpo de la solicitud de envío de datos para guardar las mediciones de temperatura y humedad que se ve en la **Figura 3.17** y la **Figura 3.18** respectivamente.

Figura 3.17. Creación de la medición de temperatura en el API.

Figura 3.18. Creación de la medición de humedad en el API.

Para la última prueba de integración se necesita mostrar en la aplicación Medición las mediciones capturadas por la red IoT además del umbral y al contenedor al que pertenecen como se ve en la **Figura 3.19**.



Fecha	Tipo Medicion	Valor Medicion	Estado	Umbral Minimo	Umbral Maximo	Contenedor
2024-02-17T18:19:23.73714...	Humedad	41.0	Normal	10.0	60.0	A20
2024-02-17T18:19:23.73714...	Temperatura	-9.4	Alerta	2.0	10.0	A20
2024-02-17T18:15:23.73216...	Humedad	40.0	Normal	10.0	60.0	A20
2024-02-17T18:15:23.73216...	Temperatura	-8.1	Alerta	2.0	10.0	A20
2024-02-17T18:07:23.72332...	Humedad	69.5	Alerta	10.0	60.0	A20
2024-02-17T18:07:23.72332...	Temperatura	-3.5	Alerta	2.0	10.0	A20
2024-02-17T17:59:28.48451...	Humedad	76.5	Alerta	10.0	60.0	A20
2024-02-17T17:59:28.48451...	Temperatura	-4.1	Alerta	2.0	10.0	A20
2024-02-17T17:45:23.68393...	Humedad	79.5	Alerta	10.0	60.0	A20
2024-02-17T17:45:23.68393...	Temperatura	-5.4	Alerta	2.0	10.0	A20

Figura 3.19. Mostrar la información de las mediciones asociadas a un contenedor y a un umbral.

3.5 Cuarta Fase: Gestión

Esta fase consiste en implementar funcionalidades de seguridad para el API, como se mencionó en la sección 2.7.3 es necesario implementar una forma de autenticación para realizar las operaciones del API. En la línea 9 del **Código 3.3** se puede notar que existe un *Middleware* para autenticar que se trata de un usuario autorizado, y se especifica en el **Código 3.4**.

```
 1. import { Router } from "express";
 2. import { authMiddleware } from "../middleware/authMiddleware.js";
 3. import { medicionController } from "../controllers/medicionController.js";
 4. import { validarDatosObtenerMediciones, validarDatosIngresarMedicion } from
"../middleware/medicionMiddleware.js";
 5.
 6. const router = Router();
 7.
 8. router.get("/medicion", authMiddleware, validarDatosObtenerMediciones ,
medicionController.obtenerTodasMediciones)
 9. router.post("/medicion", authMiddleware,validarDatosIngresarMedicion,
medicionController.crearMedicion)
10.
11. export default router;
12.
```

Código 3.3. Rutas para acceder a las operaciones con las mediciones.

```
 1. import jwt from 'jsonwebtoken';
 2. import { usuarioModel } from '../models/usuarioModel.js';
 3. import { JWT_SECRET } from "../config.js";
 4. import { check, validationResult } from "express-validator";
 5. import { verificarLlaveForanea } from '../helpers/utils.helpers.js';
 6. export const authMiddleware = async (req, res, next) => {
 7.     try {
 8.         const token = req.headers.authorization.split(' ')[1];
 9.         const decoded = jwt.verify(token, JWT_SECRET);
10.         const usuario = await usuarioModel.findByPk(decoded.id);
11.
12.         if (!usuario) {
13.             throw new Error('Usuario no encontrado');
14.
15.         }
16.
17.         req.usuario = usuario;
18.         next();
19.     } catch (error) {
20.         res.status(401).json({ error: 'Token inválido o no proporcionado' });
21.     }
22. }
23.
```

Código 3.4. Método para autenticar el token de un usuario.

El resultado es obtener la información siempre que exista un token de seguridad asociado a una sucursal, los usuarios pueden guardar la información solo de su sucursal respetando el enfoque multi arrendatario. De igual forma para la consulta de la información se realiza solo para una sucursal y con un token válido como se ve en la **Figura 3.20**.

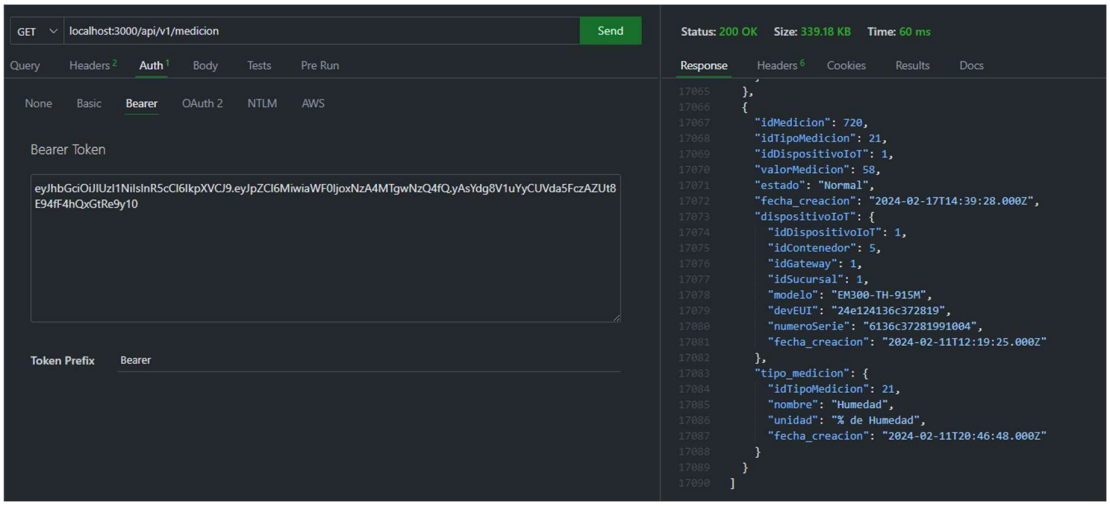


Figura 3.20. Consulta de la información de las mediciones con un token de autorización.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

El desarrollo de este proyecto conllevo al descubrimiento de las tecnologías aplicadas en los laboratorios clínicos para aumentar sus características de *Smart Lab* y aplicar el concepto de *free hands*, entre las tecnologías exploradas se encuentran: la realidad aumentada, los asistentes de voz y la tecnología IoT. En particular la instauración de la tecnología IoT permite automatizar el proceso de control de los parámetros ambientales dentro de un espacio cerrado para conocer el estado de dichos parámetros en tiempo real.

Dentro de un laboratorio clínico se ha optimizado el proceso de registro de las mediciones de temperatura y humedad en los refrigeradores y congeladores donde se almacenan los reactivos, muestras clínicas y otros insumos importantes para el desarrollo de las actividades de un laboratorista, con la instauración de una red LoRaWAN de dispositivos IoT que sigue el diagrama de despliegue de la **Figura 3.1**. Esta red de dispositivos IoT reduce el tiempo entre mediciones que se solía hacer 3 veces por día (véase en el Anexo I), a capturar una medición cada 10 minutos en promedio. Esta forma de trabajo mejora la eficiencia en el seguimiento de la temperatura y humedad de un contenedor como se ve en la **Figura 3.19**, lo que reduce significativamente el esfuerzo requerido para realizar dicha tarea por parte del laboratorista y observar los eventos que capturados por la red IoT en la aplicación de escritorio del componente Medición.

Finalmente, la información obtenida por la red de dispositivos IoT es enviada a el API por medio de una solicitud post y sigue el proceso que se observa en el diagrama de secuencia web en la **Figura 3.5**. Este proceso crea un registro de medición asociado a un dispositivo IoT y está asociada a un contendor que tiene que cumplir un umbral para determinar su estado.

4.2 Recomendaciones

Ya que el desarrollo seguido se basa en la especificación de un API antes de implementarla como se revisó en la sección 2.7 es necesario llevar un continuo control de los requerimientos que el cliente muestra, es invaluable tener *kick offs meetings* de alto valor para documentar y definir los objetivos que se persigue y los resultados, mejorando la especificación de los requerimientos que se vio en la **Tabla 3.3** y la **Tabla 3.4** respectivamente. Ya que estos son la base para el desarrollo del proyecto y evitar la pérdida de tiempo por malentendidos.

Por otro lado, el desarrollo de la red LoRaWAN de dispositivos IoT integra la adquisición y configuración del sensor EM-300 TH y el *Gateway* UG63 de la marca Milesight para realizar las tareas de monitoreo de temperatura y humedad. Sus respectivos manuales de instalación y uso se encuentran desactualizados, pero cuenta con un foro donde publican la solución de problemas de sus productos y que está en constante actualización. Esto hace que su página se soporte sea una fuente de información importante para resolver falencias durante el proceso de desarrollo de la red de dispositivos IoT.

Para finalizar se recomienda explorar más opciones a las del servidor mqtt que el *Gateway* UG63 ofrece para facilitar aún más la comunicación directa entre un API y la red de dispositivos LoRaWAN IoT, sin necesidad de contar con un *Middleware*.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] Ministerio de Sanidad, «Laboratorio Clínico Central. Estándares y recomendaciones de calidad y seguridad», Ministerio de Sanidad, Servicios Sociales e Igualdad. España, 2013, Accedido: 6 de diciembre de 2023. [En línea]. Disponible en: https://www.researchgate.net/publication/305330848_Laboratorio_Clinico_Central

[2] Ruth Cano Corres y Xavier Fuentes Arderiu, «Errores en el laboratorio clínico», Ifcc.org, Milan, Italia. Accedido: 6 de diciembre de 2023. [En línea]. Disponible en: https://cms.ifcc.org/media/214854/Errores%20en%20el%20laboratorio%20cl%C3%ADnico.pdf

[3] «Laboratory Information Management Software (LIMS)», Autoscribe Informatics.

[4] S. C. Ltda, «Orión - Sistema de Gestión de Laboratorios Cl\'\inicos», Orion-labs.com.

[5] F. J. García-Peñalvo, «Desarrollo de estados de la cuestión robustos: Revisiones Sistemáticas de Literatura», Education in the Knowledge Society (EKS), vol. 23, 2022, doi: 10.14201/eks.28600.

[6] D. Carrizo y C. Moller, «Estructuras metodológicas de revisiones sistemáticas de literatura en Ingeniería de Software: un estudio de mapeo sistemático», Ingeniare. Revista chilena de ingeniería, vol. 26, 2018, doi: 10.4067/s0718-33052018000500045.

[7] Laura Arnau Sabatés y Josefina Sala Roca, «La revisión de la literatura científica: Pautas, procedimientos y criterios de calidad», Universidad Autónoma de Barcelona, abr. 2020, Accedido: 11 de agosto de 2023. [En línea]. Disponible en: https://ddd.uab.cat/pub/recdoc/2020/222109/revliltcie_a2020.pdf

[8] D. F. de Carvalho y C. C. Miers, «Process Automation and Monitoring Systems Based on IIoT Using Private LoRaWAN Networks: A Case Study of ArcelorMittal Vega Facilities», en International Conference on Internet of Things, Big Data and Security, IoTBDS - Proceedings, 2023. doi: 10.5220/0012039300003482.

[9] J. Ramírez-Faz, L. M. Fernández-Ahumada, E. Fernández-Ahumada, y R. López-Luque, «Monitoring of temperature in retail refrigerated cabinets applying iot over open-source hardware and software», Sensors (Switzerland), vol. 20, n.o 3, 2020, doi: 10.3390/s20030846.

[10] A. Gautam y V. Sharma, «IoT and Its Future Prospect: A case study on Smart Labs». enero de 2023. doi: 10.13140/RG.2.2.22910.46408.

[11] L. González, O. Sofía, D. Laguía, E. Gesto, y K. Hallar, «Internet del Futuro – Estudio de tecnologías IoT», Informes Científicos Técnicos - UNPA, vol. 12, n.o 3, 2020, doi: 10.22305/ict-unpa.v12.n3.744.

[12] «What is LoRaWAN® specification», LoRa Alliance®. Accedido: 4 de octubre de 2023. [En línea]. Disponible en: https://lora-alliance.org/about-lorawan/

[13] Pablo Baltuille, «LoRaWAN y su aportación a las tecnologías IIoT», incibe. Accedido: 17 de noviembre de 2023. [En línea]. Disponible en: https://www.incibe.es/incibe-cert/blog/lorawan-y-su-aportacion-las-tecnologias-iiot

[14] P. Chubb, «Going smart: How IoT technologies and automation can revolutionise laboratory practice», Internet of Things News. IoT Tech News, noviembre de 2019.

[15] M. I. Rosli y M. R. Ahmad, «Internet of Things Monitoring System of a Modeled Cleanroom», ELEKTRIKA- Journal of Electrical Engineering, vol. 20, n.o 3, 2021, doi: 10.11113/elektrika.v20n3.283.

[16] Edwin Fabricio Avila Cueva y Miguel Ángel Parra Ordóñes, «Desarrollo de un prototipo de red LPWAN con tecnología LoRa para la detección de intrusos en las viviendas de una zona residencial», ESCUELA POLITÉCNICA NACIONAL, QUITO, 2020.

[17] Fiona Kuan, «¿Cuál es la tecnología detrás de la frecuencia LoRa?», Moko Smart. Accedido: 20 de octubre de 2023. [En línea]. Disponible en: https://www.mokosmart.com/es/lora-frequency/

[18] LoRa Alliance, «LoRaWAN® Regional Parameters RP002-1.0.4», Estados Unidos, 2022.

[19] A. Augustin, J. Yi, T. Clausen, y W. M. Townsley, «A study of Lora: Long range & low power networks for the internet of things», Sensors (Switzerland), vol. 16, n.o 9, 2016, doi: 10.3390/s16091466.

[20] RedHat, «What is a REST API?», RedHat. Accedido: 19 de septiembre de 2023. [En línea]. Disponible en: https://www.redhat.com/en/topics/api/what-is-a-rest-api

[21] ennVee, «MuleSoft API lifecycle management», Ennvee.com. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://blog.ennvee.com/blog/mulesoft-api-management

[22] Mike Amundsen, Design and Build Great Web APIs: Robust, Reliable, and Resilient, 1.a ed., vol. 1. Raleigh, North Carolina: The Pragmatic Programmers, 2020.

[23] Julia Martins, «10 pasos para organizar la kick off meeting perfecta para tu proyecto», asana. Accedido: 21 de noviembre de 2023. [En línea]. Disponible en: https://asana.com/es/resources/project-kickoff-meeting

[24] MileSight Network Technology Co y Ltd, «Milesight», Milesight.

[25] «Eclipse Mosquitto», Eclipse Mosquitto. enero de 2018.

[26] «IntelliJ IDEA – the leading Java and Kotlin IDE», JetBrains.

[27] B. Porter, J. van Zyl, y O. Lamy, «Welcome to Apache maven», Apache.org.

[28] «XAMPP installers and downloads for Apache friends», Apachefriends.org.

[29] «About node.Js\textregistered{}», Nodejs.org.

[30] «API Documentation & Design Tools for Teams», Swagger.io.

[31] «Express - Infraestructura de aplicaciones web Node.js», Expressjs.com.

6 ANEXOS

ANEXO I

Visita Técnica

La visita técnica se llevó a cabo el día 21 de septiembre del 2023 en las instalaciones de Asistanet en su sucursal norte de Quito, con el fin de conocer los procesos internos y las actividades que llevan a cabo diariamente dicho laboratorio clínico. Este laboratorio tiene distribuidos dentro de sus instalaciones varios refrigeradores y congeladores. Donde se almacenan las muestras, reactivos y otros productos químicos o biológicos que son necesarios para realizar sus actividades. Tomando en cuenta solo la forma de almacenar de los reactivos se obtienen los siguientes detalles:

- Los refrigeradores deben estar en un rango de temperatura de 0°C a 10°C para que las muestras y los reactivos abiertos y sellados se mantengan en su período de duración hasta su caducidad.
- Los congeladores soportan temperaturas desde -20°C en adelante.
- Solo se tiene refrigeradores y congeladores donde se separan los reactivos dependiendo de sus especificaciones.
- Los dispositivos de medida deben ser certificados para que el intervalo de error no sea muy amplio.
- En la actualidad utilizan un termómetro calibrado con pruebas de precisión que son certificados por EMCO.
- La forma de identificar los refrigeradores y congeladores son numerados. Ejemplo: congelador 1, refrigerador 1, etc. Estos son identificados para llevar el registro de la temperatura en el día.
- El registro de la temperatura se lo realiza 3 veces al día indistintamente de la hora, pero se calcula cual fue la medida más alta y baja. Además, se calcula el promedio de las 3 temperaturas para registrar en una tabla. Esto mantiene el control de la temperatura de los almacenes.
- Cada uno de los insertos (reactivos) tienen su documentación de uso y almacenamiento, en él se detalla la temperatura para el almacenamiento cuando se encuentra abierto y cuando se encuentra sellado. También muestra el tiempo de duración de los reactivos dependiendo de su almacenaje.

- Extra: Existen contenedores de transporte de muestras en donde se podría probar la inclusión de dispositivos H-IoT para ver el estado de las muestras mientras son transportadas dentro del laboratorio.
- Planificar un control para evitar el fallo en la medida de la temperatura, bajo un tiempo específico o de forma aleatoria.

Anexo II

***Benchmark* de los dispositivos IoT**

Características	**Milesight EM300-TH**	**SenseCAP S2101**
Imagen		
Configuración LoRaWAN		
Frecuencia LoRaWAN	CN470/RU864/IN865/EU868/US915/AU915/KR920/AS923	IN865/EU868/US915/AU915/ AS923/KR920/RU864
Clase Dispositivo	OTAA/ABP Clase A	LoRaWAN v1.0.3 Clase A
Temperatura del aire		
Gama	-30°C a + 70°C	-40 hasta +85 °C
Exactitud	De 0 °C a + 70 °C (±0,3 °C), de -30 °C a 0 °C (±0,6 °C)	±0.2 °C
Resolución	0.1°C	0.01 °C
Humedad del aire		
Gama	0% a 100% HR	0 a 100% HR
Exactitud	10% a 90% HR (±3%), por debajo del 10% y por encima del 90% HR(±5%)	±1,8 %HR
Resolución	0.5% HR	0,01 %HR
Características de funcionamiento		
Temperatura de funcionamiento	-30°C a 70°C	-40 hasta +85 °C
Clasificación IP	IP67	IP66
Humedad de funcionamiento	0% a 100% HR	0 a 100% HR (sin condensación)
Duración batería	≧ 5 años (por 1 batería)	Hasta 10 años
Capacidad de la batería	ER18505 batería Li-SOCl2 4000 mAh (8000 mAh opcional)	19Ah (no recargable)
Distancia de comunicación	2 a 10 km	2 a 10 km (dependiendo de la antena de la puerta de enlace y los entornos)
Tecnología de configuración	NFC	Bluetooth integrado
Precio	$ 99.00 dólares	$ 59.00 dólares

***Benchmark* de los *Gateways* de la red LoRaWAN IoT**

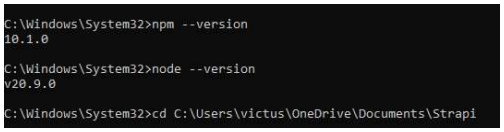
Características	**Milesight *Gateway* UG63**	**SenseCAP *Gateway* SX1320**
Imagen		
Configuración LoRaWAN		
Banda de frecuencia	AU915/AS923/US915	IN865/EU868/RU864/US915/ AU915/KR920/AS923
Antenas	2 × antenas internas	Wi-Fi: antena interna y LoRaWAN
Canales	8 (Half/Full-duplex)	8
Protocolos soportados	V1.0 Clase A/Clase B/Clase C y V1.0.2 Clase A/Clase B/Clase C	V1.0.3 Clase A/Clase B/Clase C
Software		
Protocolos de red	PPPoE, SNMP v1/v2c/v3, TCP, UDP, DHCP, DDNS, HTTP, HTTPS, DNS, ARP, SNTP, Telnet, SSH, MQTT, etc.	PPPoE, SNMP v1/v2c/v3, TCP, UDP, DHCP, DDNS, HTTP, , MQTT, etc.
Aplicación	Python SDK	
Administración	Web, CLI, DeviceHub, Milesight IoT Cloud, Yeastar Workplace Platform	Telaraña
Características físicas		
Entrada de alimentación	1. 1 × entrada PoE estándar 802.3af 2. 5V por puerto tipo C	DC 12V - 2A PoE (IEEE 802.3 af), 40V-57V DC
Protección IP	IP30	
Temperatura de funcionamiento	-20 °C a +50 °C (-4 °F a +122 °F)	-20°C a 55°C
Entradas	Ethernet, USB C	Ethernet RJ45 * 1, Conector de antena hembra RP-SMA * 1, Ranura para tarjeta Micro SD (uso futuro) * 1, USB tipo C (consola) *
Precio	$299.00 dólares	$99.00 dólares

Anexo III

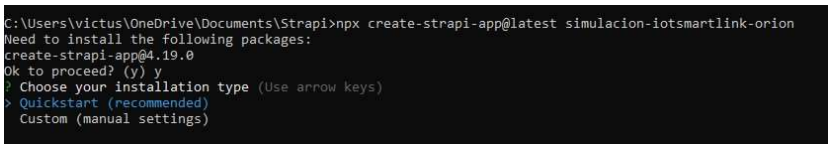
Instalación de Strapi

Strapi es un sistema de gestión de contenidos (CMS) de código abierto que permite a los desarrolladores la libertad de construir un API con las características REST, sin importar la tecnología que deseen utilizar. Es flexible ya que permite la personalización en la forma de abordar un caso de uso o requerimiento. También consta de formas de autenticación de usuarios por roles para asemejarse a el desarrollo real de un API REST.

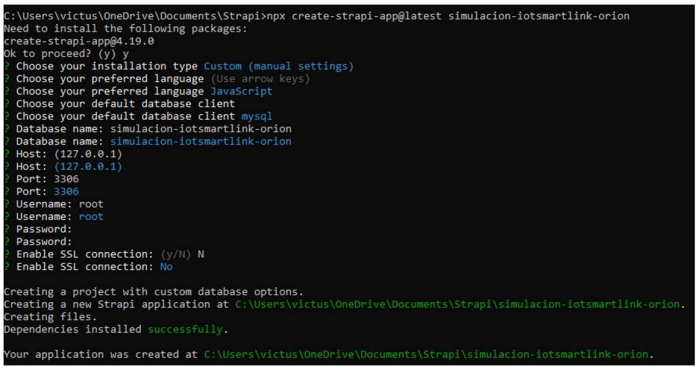
Para utilizar Strapi es necesario tener instalado dos herramientas fundamentales: npm y node.js. Para comprobar su instalación se lo realiza en la línea de comandos.



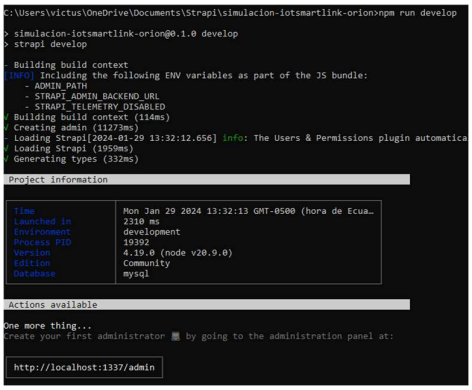
El siguiente paso consta de crear un proyecto de Strapi con un nombre.



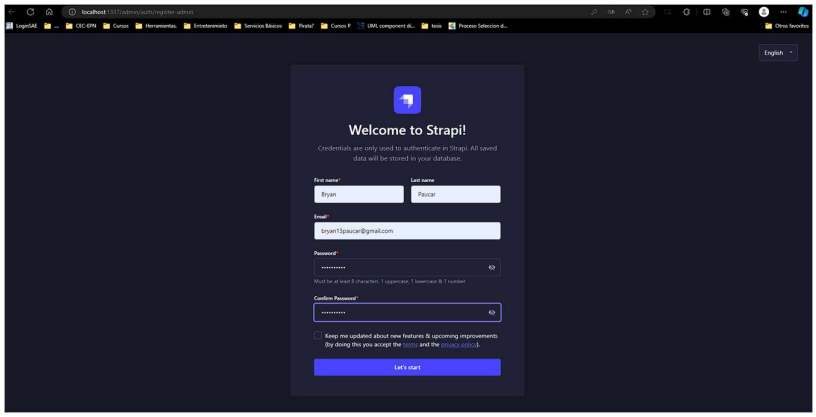
Es necesario realizar una instalación manual para poder conectar el proyecto de Strapi con la base de datos MySQL. Y las configuraciones posteriores que se nos pide para poder realizar esta conexión a la base de datos.



Una vez concluida la instalación es necesario ponerla a ejecutar para comprobar si no existió algún problema en la configuración e instalación.



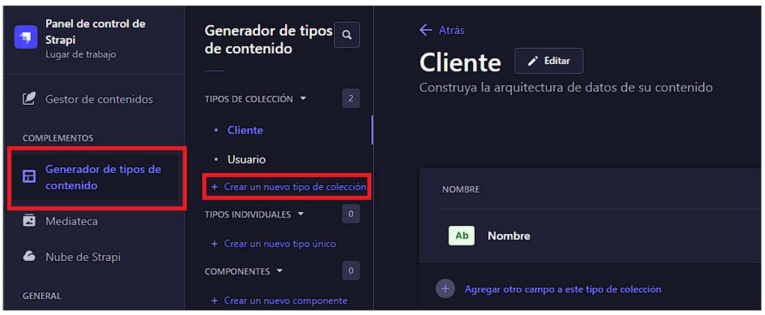
Para finalizar se comprueba en la dirección otorgada por la ejecución de Strapi si se puede acceder a el proyecto creado.



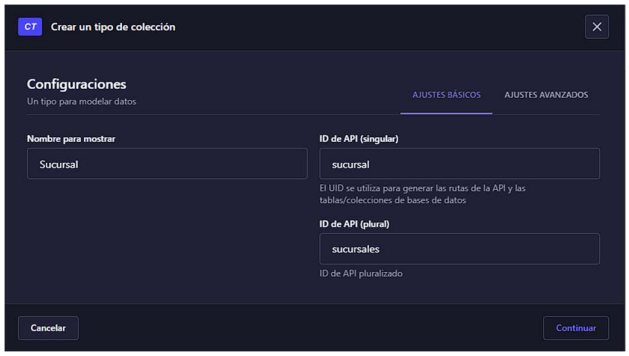
Creación de la simulación del API IoTSmartLink-Orion

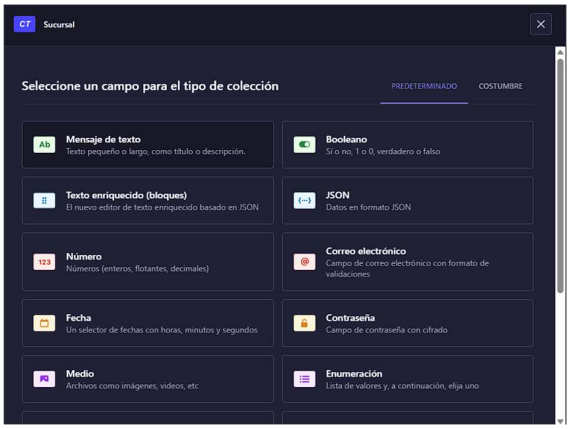
Creación de colecciones y campos del API

Lo primero para generar el API es crear las colecciones de datos, estas colecciones funcionan como las tablas de una base de datos relacional. Para crear estas colecciones se debe ir al apartado generador de tipos de contenido y seleccionar crear una nueva colección.

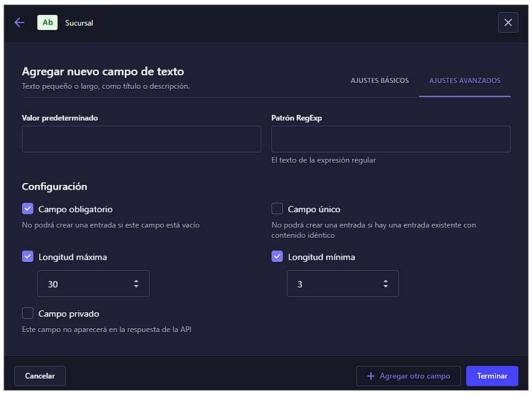


Al crear una nueva colección es necesario ingresar su nombre y los campos de datos que se encuentran dentro de esta colección. Strapi permite seleccionar múltiples tipos de campos dependiendo de las necesidades del desarrollador.

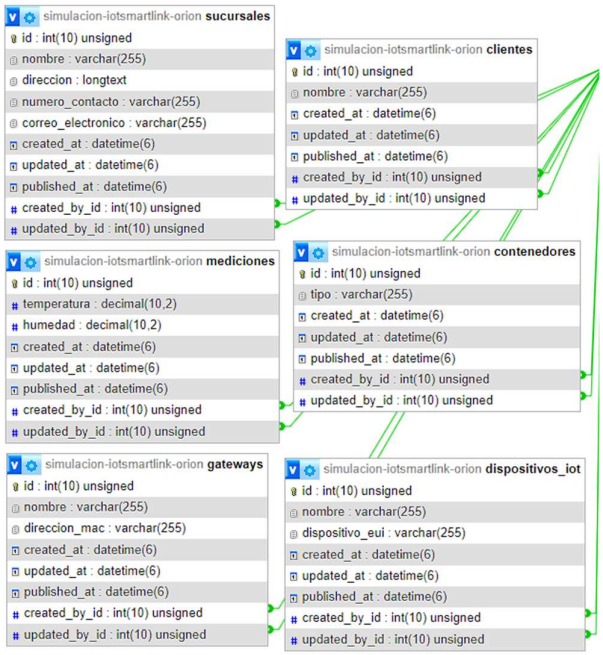




El siguiente paso es darle un nombre al capo que se desea crear, además, se realizan una configuración de campo obligatorio y definir un máximo y mínimo número de caracteres. Cabe destacar que cada uno de los tipos de datos cuentan con sus configuraciones de tamaño y de obligatoriedad.



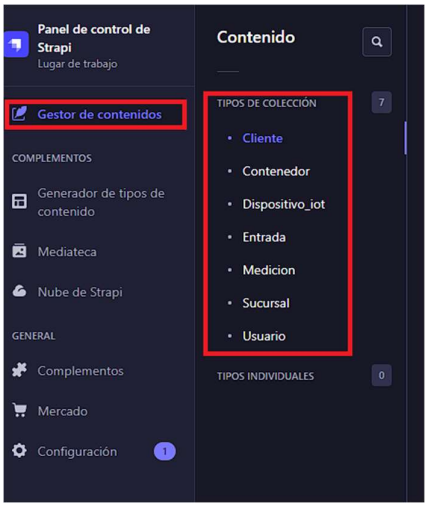
Al final el campo nombre se ha creado dentro de la colección sucursal y se repite el mismo proceso dependiendo de la cantidad de campos que se deseen crear por cada colección. El resultado de crear las colecciones y sus campos, donde se observa la estructura de las tablas que servirán para la simulación del API. Cabe destacar que es necesario tener del registro de *Gateway* asociado a una sucursal y está asociada a un laboratorio clínico para obtener la información. Es por ello por lo que fue necesario crear las colecciones *Gateway*, cliente y sucursal.



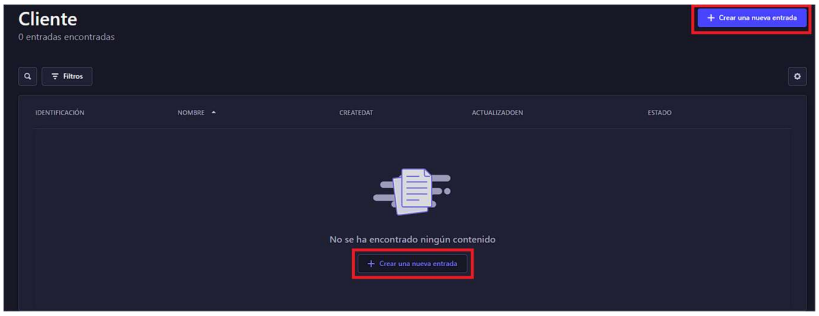
Cabe destacar que Strapi genera otros campos aparte de los creados por el usuario, estos campos son *created_at*, *updated_at*, *published_at*, *created_by_id* y *updated_by_id*. Dichos campos sirven para mantener el control de los cambios que se realice dentro del API, lo cual también ayuda a tener una idea de cómo quedarían las tablas para el registro de la información.

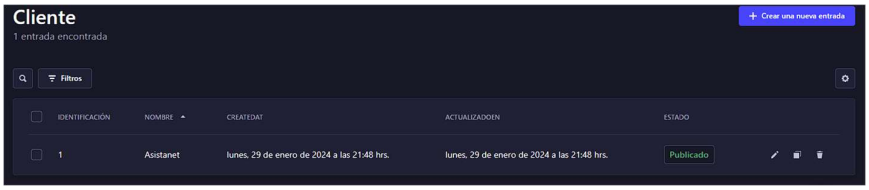
Añadir datos a los campos de las colecciones

Para que poder simular el API es necesario que los campos de las colecciones tengan datos para poder ser consultados, se necesita tener información de un laboratorio, su sucursal y un *Gateway* asociado. Strapi permite la ingresar datos de forma sencilla para hacerlo se debe seleccionar en el panel de control el gestor de contenidos y seleccionar una de las colecciones.



Posteriormente se crea una nueva entrada en la colección cliente y se añade los campos configurados de cada colección. El ingreso de los datos para el cliente una vez realizado el ingreso se debe seleccionar salvar para guardar la información y después publicar para poner esa información disponible.

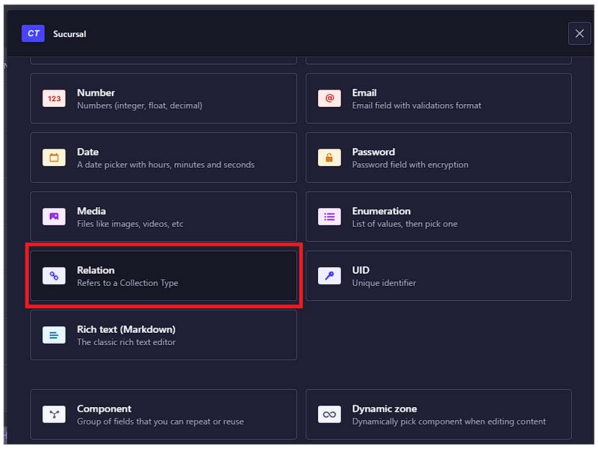




Todo este proceso se realiza para cada una de las colecciones y sus atributos, ya con datos dentro de la base de datos solo queda realizar las relaciones para que se complete el modelo relacional para la simulación de API *IoTSmartLink-Orion*.

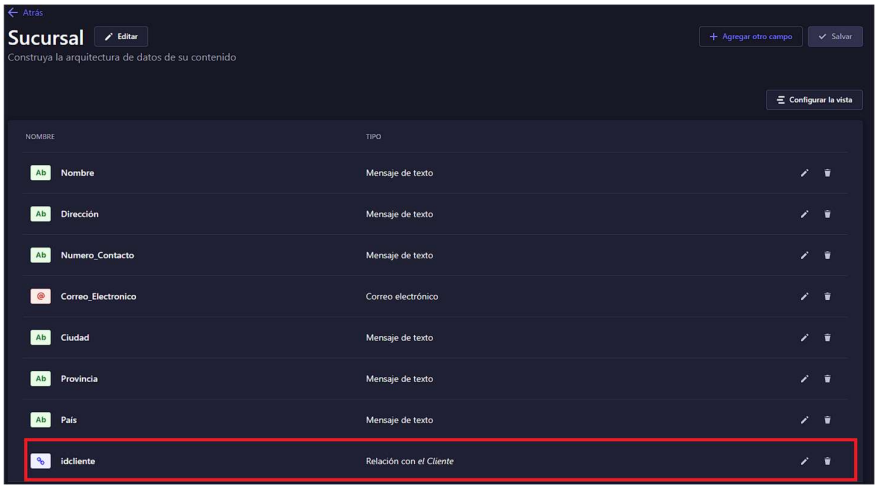
Relaciones entre las colecciones

Una vez creadas las tablas o colecciones es necesario relacionarlas entre ellas para conformar así un modelo relacional de base de datos. Strapi permite realizar este proceso definiendo el tipo de cardinalidad que se desea aplicar: uno a uno, uno a muchos y muchos a muchos.



El siguiente paso es seleccionar la cardinalidad que se desea implementar, Strapi permite utilizar diferentes cardinalidades y pone un texto de ayuda para comprobar cómo se entendería la relación entre las colecciones.

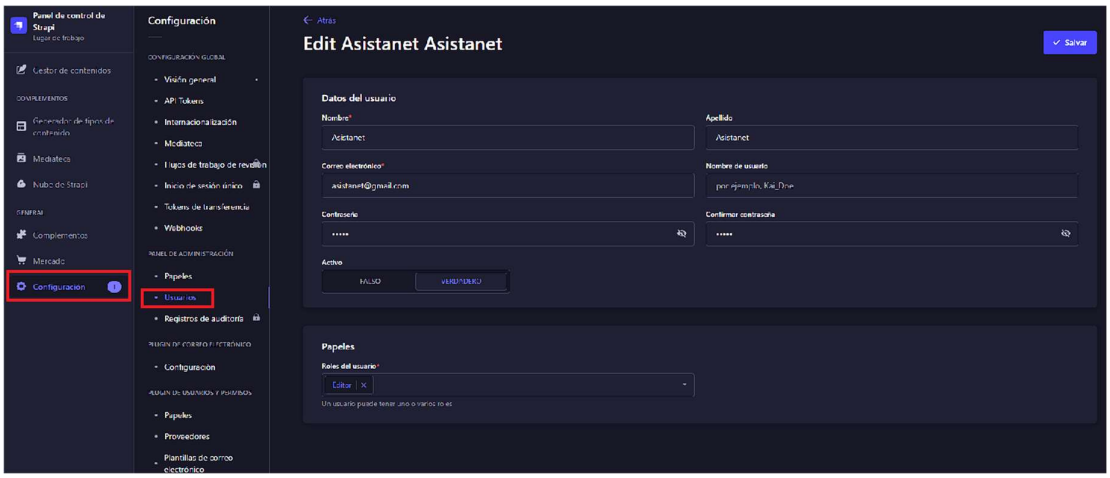
Se guardó el nuevo campo de relación con otra colección. Este proceso se repite para todas las colecciones para obtener un diagrama entidad relación de la simulación del API *IoTSmartLink-Orion*.



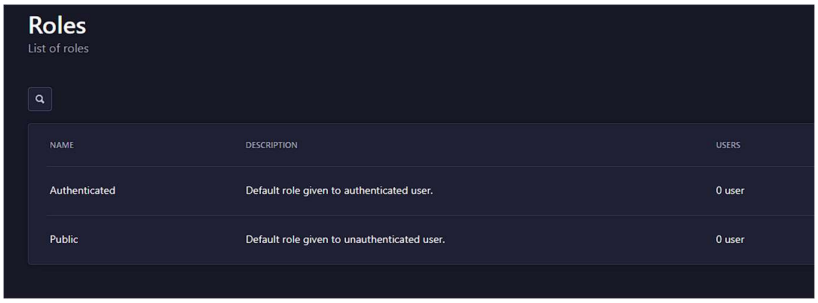
Una vez terminada las relaciones es necesario actualizar todos los datos ingresados en cada una de las colecciones para asociarlos con las diferentes relaciones y así obtener datos relacionados y listos para utilizarlos en el API.

Generación de los *end points* del API

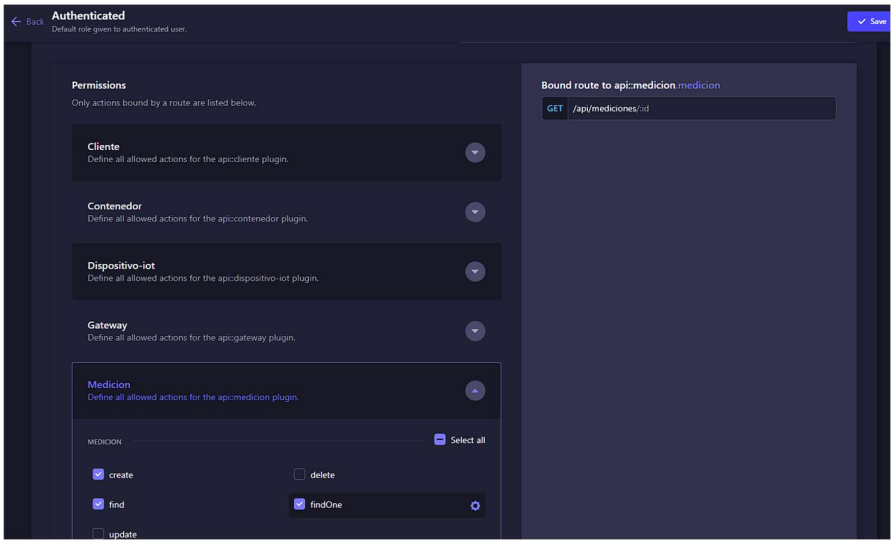
Para generar los *end points* del API es necesario seguir el diagrama de secuencia web, donde se especifica que al iniciar la sesión en el API se genera el token para las consultas y cada consulta debe estar acompañada por dicho token. Para ello es necesario crear usuarios de un laboratorio y darles acceso a la información.



Este paso es necesario ya que Strapi por defecto pide autenticación de algún usuario para poder realizar consulta o enviar información. Una vez creado el usuario se le da los permisos necesarios en el apartado configuración>roles, estos roles definen el acceso al API.



Dentro de los roles Autenticados se crea los *end points* para que un usuario autenticado con un token de acceso pueda realizar las operaciones de consulta y de envió de información. Las colecciones se puede habilitar todas las funcionalidades de un API REST, pero en este caso solo interesa activar la operación de consulta para las colecciones: dispositivosIoT y contenedores. Mientras que para las mediciones se activa las operaciones de envión y consulta de la información.



Para finalizar se guarda y se han creado los *end points* de simulación del API IoTSmartLink-Orion. Esta API de simulación ya puede ser utilizada.

Anexo III

```
alps:
  version: '1.0.0'
  title: 'API IoTSmartLink-Orion'
  doc:
    type:"markdown"
    value:"Este es el ALPS para el API IoTSmartLink-Orion"
  descriptor:
  ##Vocabulario de propiedades
    - id: "usuario"
      type: "semantic"
    - id: "contraseña"
      type: "semantic"
    - id: "token"
      type: "semantic"
    - id: "idDispositivoIoT"
      type: "semantic"
    - id: "idMedicion"
      type: "semantic"
    - id: "idContenedor"
      type: "semantic"
    - id: "fecha"
      type: "semantic"
    - id: "temperatura"
      type: "semantic"
    - id: "humedad"
      type: "semantic"
  ##Vocabulario de relaciones
  "Home":
    - id: "Home"
      type: "group"
      descriptor:
        - "href": "#usuario"
        - "href": "#contraseña"
        - "href": "#token"
        - "href": "#idDispositivoIoT"
        - "href": "#idMedicion"
        - "href": "#idContenedor"
        - "href": "#fecha"
        - "href": "#temperatura"
        - "href": "#humedad"
  ##Acciones
  actions:
  #Acceso a la información del usuario autenticado
    - id: "obtenerToken"
      name: "obtenerTokenUsuario"
      type: "safe"
      descriptor:
        - "href": "#usuario"
        - "href": "#contraseña"
  #Obtener un dispositivo IoT
    - id: "obtenerDispositivoIoT"
      name: "obtenerInformacionDispositivoIoT"
      type: "safe"
      descriptor:
        - "href": "#idDispositivoIoT"
        - "href": "#token"
  #Obtener las Mediciones
    - id: "obtenerMediciones"
      name: "obtenerInformacionMediciones"
      type: "safe"
      descriptor:
        - "href": "#idDispositivoIoT"
        - "href": "#token"
        - "href": "#fecha_creacion"
        - "href": "#idMedicion"
  #Obtener un Contenedor
    - id: "obtenerContenedor"
      name: "obtenerInformacionContenedor"
```

```
      type: "safe"
      descriptor:
        - "href": "#idContenedor"
        - "href": "#token"
  #Guardar Medidiones
    - id: "guardarMediciones"
      name: "guardarInformacionMediciones"
      type: "safe"
      descriptor:
        - "href": "#idDispositivoIoT"
        - "href": "#token"
        - "href": "#temperatura"
        - "href": "#humedad"

```

Anexo IV

Manual de usuario del *Gateway* UG63 de Milesight:

https://resource.milesight.com/milesight/iot/document/ug63-user-guide-en.pdf

Manual de usuario del sensor EM-300 TH:

https://resource.milesight.com/milesight/iot/document/em300-series-user-guide-en.pdf

Instalación del bróker Mosquitto:

https://ine4celectronics.com/wp-content/uploads/2021/01/INSTALACION-BROKER-MOSQUITTO-WINDOWS-RASPBERY.pdf

Anexo V

Enlace al repositorio de las pruebas:

https://github.com/BPaucar01/Pruebas-del-API

Enlace al repositorio del desarrollo del API:

https://github.com/BPaucar01/API-IoTSmartLink-Orion

Enlace al repositorio del desarrollo de la aplicación de escritorio:

https://github.com/BPaucar01/MedicionesService.git