|  |  |  |
| --- | --- | --- |
|  | Universidad de La Serena  Escuela de Ingeniería en Computación |  |

MEMÓRIA DE TÍTULO

INGENIERO EN COMPUTACIÓN

**LIITEC API: Sistema para recolección y procesamiento de datos medidos de sensores ambientales**

Alumno

ANTONY MAXIMILIANO RODRÍGUEZ CARHUACHIN

Profesor Guía

TOMÁS CARVAJAL ROJAS

Profesor Patrocinante

FRANCISCO LOPEZ

La Serena, 2023

Resumen

El presente trabajo tiene como objetivo…

Agradecimientos

Agradezco a …..

Tabla de contenido

[1. Introducción 1](#_Toc158321178)

[1.1. Motivación 2](#_Toc158321179)

[1.2. Meta 2](#_Toc158321180)

[1.3. Objetivo General 2](#_Toc158321181)

[1.4. Objetivos Específicos 3](#_Toc158321182)

[1.5. Ámbito 3](#_Toc158321183)

[1.6. Alcance 3](#_Toc158321184)

[2. Marco Teórico 4](#_Toc158321185)

[2.1. Contexto Tecnológico y Fundamentos 4](#_Toc158321186)

[2.1.1. Recolección de Datos en Entornos Científicos 4](#_Toc158321187)

[2.1.2. Plataformas de Terceros y sus Limitaciones 5](#_Toc158321188)

[2.1.3. Limitaciones Comunes en Plataformas de Terceros 6](#_Toc158321189)

[2.1.4. Relevancia de las API en el Ámbito Tecnológico 7](#_Toc158321190)

[2.2. Herramientas de Recolección de Datos 9](#_Toc158321191)

[2.3. Arquitectura y Tecnologías Utilizadas 10](#_Toc158321192)

[2.3.1. Arquitectura 10](#_Toc158321193)

[2.4. Tecnologías Empleadas 13](#_Toc158321194)

[2.4.1. API web REST 14](#_Toc158321195)

[2.4.2. Docker 17](#_Toc158321196)

[2.4.3. Broker MQTT – Eclipse Mosquitto 18](#_Toc158321197)

[2.4.4. Express 20](#_Toc158321198)

[2.4.5. MongoDB 21](#_Toc158321199)

[2.4.6. Amazon Web Services EC2 23](#_Toc158321200)

[3. Arquitectura Final de LIITEC API 23](#_Toc158321201)

[4. Bibliografía 24](#_Toc158321202)

# Introducción

En los últimos años, Chile ha enfrentado desafíos climáticos sin precedentes que han impactado profundamente en diversas regiones del país. El cambio climático ha exacerbado la frecuencia y la intensidad de eventos meteorológicos extremos, tales como sequías prolongadas y olas de calor, que afectan no solo al entorno natural, sino también a la vida cotidiana de las comunidades. En este contexto, la Región de Coquimbo, situada en el norte de Chile, ha experimentado una de las sequías más graves de las últimas décadas [1], con efectos devastadores en la disponibilidad de agua y en la economía regional.

Según informes del Ministerio del Medio Ambiente y la Dirección Meteorológica de Chile [2], el país enfrenta la perspectiva de sumar 14 años consecutivos de sequía, lo que evidencia los impactos tangibles del cambio climático en la región. Además, se proyecta un aumento en la frecuencia de eventos climáticos extremos, como olas de calor y marejadas, lo que representa un desafío adicional para la gestión ambiental y la adaptación a estos fenómenos.

En este contexto adverso, la implementación de tecnologías para la recolección de datos ambientales cobra una importancia significativa. La recopilación y análisis de información proveniente de sensores ambientales se convierte en una herramienta fundamental para comprender y monitorear el impacto del cambio climático en la región, así como para desarrollar estrategias de adaptación y mitigación.

El Laboratorio de Investigación e Innovación Tecnológica para la Educación en Ciencias (LIITEC) se encuentra inmerso en este contexto, donde la obtención precisa y oportuna de datos desempeña un papel crucial en sus actividades de investigación y desarrollo. Una de los objetivos de LIITEC es colaborar con la recolección de datos para la toma de decisiones informada así poder sobrellevar los efectos del cambio climático. Desarrollar un sistema para la recolección de datos de sensores ambientales, sería una solución acertada. A través de la recopilación de información proveniente de la zona y la implementación de tecnologías innovadoras, este sistema busca contribuir al monitoreo y la comprensión de los cambios ambientales en la región, así como proporcionar herramientas para la toma de decisiones informadas en materia de gestión ambiental y desarrollo sostenible.

En los próximos capítulos se detallará los componentes y el proceso para llevar a cabo este sistema, destacando cómo esta solución se convierte en una herramienta valiosa para abordar los efectos del cambio climático en la región. Además, se analizará cómo el desarrollo de este sistema puede ser una contribución significativa para enfrentar los desafíos ambientales a nivel local.

## Motivación

Este proyecto surge en el marco de una de las prácticas de la carrera, durante una conversación con el supervisor de la práctica en la que se abordaban temas sobre la fabricación de estaciones de medición de variables ambientales, la forma en como los datos son transferidos y en donde estos eran almacenados. En ese contexto, se hizo énfasis en el alojamiento de los datos medidos por los sensores, revelándose el uso de ThingSpeak, un servicio de análisis de IoT (Internet de las Cosas) que facilita la agregación, visualización y análisis de flujos de datos en tiempo real en la nube[[1]](#footnote-1).

Aunque hasta ahora esta plataforma ha demostrado eficiencia en el almacenamiento y visualización de datos, ha surgido una inquietud crucial respecto a su idoneidad a largo plazo para los objetivos del Laboratorio LIITEC, uno de ellos el monitoreo del aumento temperatura a causa de las olas de calor [3]. Al profundizar en la investigación, se ha encontrado respaldo en la literatura que destaca los riesgos asociados con depender exclusivamente de plataformas de terceros, en términos de flexibilidad y adaptabilidad, como se expone en el artículo "*Third-party apps: what are the risks?*” de bcs365 [4].

El artículo aborda de manera significativa los peligros vinculados a las aplicaciones de terceros, resaltando la preocupación por la seguridad cibernética, la posibilidad de fugas de datos, la complejidad de integración, y la falta de transparencia por parte de los proveedores. Específicamente, se destaca el riesgo de malware, la preocupación por fugas de datos, la complejidad en la integración, la falta de transparencia de los proveedores, y los problemas potenciales de seguridad y compatibilidad debido al abandono del soporte por parte de los desarrolladores de aplicaciones de terceros.

La principal motivación detrás de este proyecto es abordar los riesgos identificados y crear una solución interna que garantice tanto el eficiente almacenamiento de los datos recolectados como un mayor control sobre su gestión y aplicación. En el entorno del Laboratorio LIITEC, donde la precisión y confiabilidad de los datos son fundamentales para la investigación educativa en ciencias, la implementación de una Interfaz de Programación de aplicaciones (API) propia se ve como una oportunidad práctica y aplicativa de los conocimientos adquiridos durante la formación en la carrera. Asimismo, este proyecto ofrece la perspectiva de contribuir de manera significativa al avance de la investigación educativa y al monitoreo y comprensión de los cambios ambientales en la región, al proporcionar una herramienta personalizada que aborda los riesgos y desafíos previamente señalados en la literatura especializada.

## Meta

Optimizar la gestión de datos de los sensores ambientales en LIITEC, facilitando su procesamiento y utilización efectiva en la toma de decisiones ambientales.

## Objetivo General

Desarrollar e implementar una API personalizada para la recolección de datos de sensores ambientales, con el propósito de optimizar la eficiencia en el almacenamiento de datos y ofrecer mayor control sobre la gestión de la información en LIITEC. Esta API se diseñará con énfasis en la adaptabilidad a los requisitos de gestión de datos de sensores ambientales, buscando proporcionar una solución técnica sólida y centrada en la eficacia del procesamiento de datos y la toma de decisiones ambientales.

Desarrollar e implementar una API personalizada para la recolección de datos en LIITEC, con el propósito de optimizar la eficiencia en el almacenamiento de datos y ofrecer mayor control sobre la gestión de la información. Esta API se diseñará con especial énfasis en la adaptabilidad a las necesidades específicas del laboratorio, buscando así proporcionar una solución técnica sólida y centrada en los requerimientos del usuario final.

## Objetivos Específicos

1. Evaluar el tipo de API más adecuado para la arquitectura, considerando los requerimientos de recolección y gestión de datos.
2. Diseñar una arquitectura de API segura y escalable que garantice la confidencialidad y disponibilidad de los datos.
3. Aplicar medidas de seguridad para proteger la integridad y confidencialidad de la información recolectada.
4. Implementar un prototipo funcional con capacidades eficientes de recolección y gestión de datos.
5. Desarrollar un módulo de integración específico para los sensores utilizados por LIITEC, asegurando su compatibilidad y correcto funcionamiento dentro de la API.
6. Diseñar e implementar una interfaz de usuario intuitiva que facilite la interacción con la API y sus funcionalidades.
7. Documentar detalladamente la API, proporcionando una guía clara sobre su funcionamiento, uso y mantenimiento.
8. Realizar pruebas para validar la funcionalidad, seguridad y rendimiento de la API, garantizando su fiabilidad y eficacia en la gestión de datos ambientales en LIITEC.
9. Identificar los requisitos específicos del Laboratorio LIITEC para la recolección y gestión de datos ambientales.
10. Evaluar diferentes tecnologías y herramientas disponibles para el desarrollo de una API que cumpla con los requisitos identificados.
11. Diseñar y desarrollar una arquitectura de API escalable y segura, que garantice la disponibilidad y confidencialidad de los datos recolectados.
12. Implementar un prototipo funcional de la API con características para la recolección y gestión eficiente de datos ambientales.
13. Desarrollar un módulo de integración específico para los sensores utilizados por LIITEC, asegurando su compatibilidad y correcto funcionamiento dentro de la API.
14. Implementar medidas de seguridad para proteger la integridad y confidencialidad de la información recolectada.
15. Diseñar e implementar una interfaz de usuario para la gestión de datos, que permita a los usuarios de LIITEC interactuar de manera eficiente con la API y sus funcionalidades.
16. Documentar la API, proporcionando una guía clara y detallada sobre su funcionamiento, uso y mantenimiento, para facilitar su comprensión y futuras actualizaciones.
17. Realizar pruebas para validar la funcionalidad, seguridad y rendimiento de la API, garantizando su fiabilidad y eficacia.

## Ámbito

Aca va el ambito

## Alcance

Aca va el alcance

Marco Teórico

La presente sección constituye el cimiento conceptual y metodológico fundamental que permite una comprensión profunda del contexto en el que se enmarca este proyecto. Para facilitar la asimilación de la información, esta sección se estructura en dos componentes esenciales: el contexto tecnológico y metodológico, y las tecnologías específicas que desempeñan un papel clave en el desarrollo de la API.

## Contexto Tecnológico y Fundamentos

En esta sección exploraremos el entorno tecnológico que rodea la recolección de datos en entornos científicos, examinando las plataformas de terceros y sus limitaciones, así como la relevancia crucial de las APIs en el ámbito tecnológico.

### Recolección de Datos en Entornos Científicos

La recolección de datos en entornos científicos ha experimentado una transformación significativa gracias a los avances tecnológicos, como se evidencia en el artículo "*The technological advancements that enabled the age of big data in the environmental sciences: A history and future directions*" [5]. El documento anterior proporciona una visión detallada de la evolución de la recolección de datos en ciencias ambientales y destaca cómo la tecnología ha desempeñado un papel fundamental en este proceso. A continuación, se profundiza en los puntos clave presentados en el artículo:

1. Evolución de Instrumentación y Sensores:

Antes de los desarrollos tecnológicos actuales, la recolección de datos en ciencias ambientales se basaba en evidencia visual y pruebas químicas poco específicas, con límites de detección elevados. La limitada capacidad de recopilación de datos y la participación humana necesaria imponían restricciones significativas a la comprensión completa de los fenómenos ambientales.

Con el tiempo, la instrumentación mejorada, que incluye sensores ambientales, secuenciadores genéticos y otros instrumentos analíticos, ha permitido a los investigadores abordar preguntas ambientales complejas de manera más precisa y eficiente. La capacidad de tomar muestras en el campo, registrar ubicaciones mediante dispositivos móviles y analizar muestras con instrumentos analíticos avanzados ha aumentado significativamente, permitiendo una recolección de datos más completa y detallada.

1. Enfoque en Big Data en Ciencias Ambientales:

La llegada del término "Big Data[[2]](#footnote-2)" en 2005 marcó un cambio significativo en la forma en que se aborda la recolección y análisis de datos. Big Data se caracteriza por sus "cuatro V": volumen, velocidad, variedad y veracidad. Este enfoque se ha vuelto fundamental para gestionar conjuntos de datos ambientales cada vez más grandes y complejos.

1. Digitalización y Sensores Remotos:

Los avances en la digitalización de datos y la capacidad de realizar mediciones a través de sensores remotos han sido cruciales para la era de Big Data en ciencias ambientales. Desde cámaras montadas en aviones hasta sensores en satélites, UAVs y estaciones fijas, la accesibilidad a datos multiespectrales e hiperespectrales ha aumentado, permitiendo la monitorización de cambios en el uso del suelo, la identificación de fuentes históricas de contaminación y la evaluación de nutrientes en plantas

1. Mejora en Sensores Ambientales:

Los sensores ambientales han evolucionado desde dispositivos analógicos y basados en papel hasta instrumentos digitales que registran y transmiten datos de manera eficiente y precisa. Estos avances han permitido la toma de datos en intervalos específicos con mínima intervención humana, mejorando la eficiencia y reduciendo los costos de producción de sensores.

1. Biosensores y Análisis en el Campo:

Los avances en tecnología han permitido llevar a cabo análisis ambientales directamente en el terreno mediante el uso de biosensores. Esta innovación ha implicado una reducción de costos y un aumento en la eficiencia en la recopilación de datos, al eliminar la necesidad de equipos especializados y laboratorios centralizados.

En resumen, la recolección de datos en entornos científicos ha evolucionado de manera significativa gracias a la instrumentación avanzada y los sensores mejorados. La capacidad de capturar grandes volúmenes de datos de manera eficiente y precisa ha allanado el camino para el enfoque de Big Data en ciencias ambientales, proporcionando nuevas oportunidades para comprender y abordar desafíos ambientales complejos.

### Plataformas de Terceros y sus Limitaciones

La recolección de datos de sensores ha experimentado una transformación significativa gracias a la proliferación de plataformas especializadas que facilitan la adquisición, almacenamiento y análisis de información proveniente de dispositivos sensoriales. Estas plataformas, que van más allá de ser simplemente herramientas, desempeñan un papel integral en la investigación científica, la monitorización ambiental y proyectos de IoT. A continuación, se presentan algunas plataformas destacadas junto con sus características y desafíos asociados.

1. ThingSpeak

Es un servicio de plataforma de análisis de IoT que tiene como objetivo agregar, visualizar y analizar flujos de datos en vivo en la nube. Esta solución de software está diseñada para sistemas IoT pequeños y medianos, así como para organizaciones y empresas más grandes. Según el proveedor, ThingSpeak es utilizado por profesionales de IoT, analistas de datos, ingenieros, investigadores y aquellos en el campo del monitoreo ambiental.

Limitaciones:

* Restricciones en el número de canales y campos en planes gratuitos.
* Funcionalidades avanzadas como la escalabilidad y soporte técnico requieren suscripciones pagas.[[3]](#footnote-3)

1. Ubidots

Es una plataforma de IoT que permite a las empresas crear aplicaciones IoT y convertir los datos de los sensores en conocimiento práctico y aplicable. Es una herramienta que facilita la transformación digital al proporcionar visualizaciones, alertas inteligentes, reportes programados, análisis de datos y más herramientas para tomar decisiones basadas en datos.

Limitaciones:

* Carece de un plan gratuito, sin embargo, ofrece una prueba de 30 días.
* Ofrece almacenamiento en la nube para los datos de los sensores, pero puede haber limitaciones en la capacidad de almacenamiento dependiendo del plan o suscripción utilizada.
* Limitaciones en la capacidad de escalar y manejar grandes volúmenes de datos.[[4]](#footnote-4)

1. Blynk

Es una plataforma de IoT que permite crear interfaces de control personalizadas para tus proyectos IoT. Puedes controlar y monitorear dispositivos de hardware desde tu teléfono móvil, guardar y visualizar datos, y realizar acciones basadas en eventos.

Limitaciones:

* Número limitado de widgets y proyectos, así como un límite en la cantidad de notificaciones y actualizaciones de datos por segundo en su capa gratuita.
* Es adecuado para proyectos individuales y pequeñas aplicaciones, puede enfrentar desafíos de escalabilidad en proyectos más grandes o con un gran número de dispositivos conectados.
* Utiliza servicios en la nube para la comunicación entre la aplicación y los dispositivos de hardware. Esto implica una dependencia de terceros y puede haber limitaciones o interrupciones en el servicio debido a problemas en la infraestructura de la nube.[[5]](#footnote-5)

1. Losant

Plataforma empresarial de IoT que proporciona una sólida base de software en el borde y en la nube. Destaca por su enfoque único y low-code, permitiendo a los desarrolladores crear y gestionar eficientemente aplicaciones IoT. Ofrece herramientas para la conexión, visualización y control de dispositivos, así como para el procesamiento y análisis de datos en tiempo real. La agilidad y velocidad de comercialización son elementos clave de esta solución integral para aplicaciones de IoT escalables y atractivas.

Limitaciones:

* Organizaciones y entornos aislados tienen límites en dispositivos y aplicaciones, con posibles costos adicionales para ajustes.
* Restricciones en la cantidad de datos que se pueden cargar, con diferenciación entre límites mensuales para organizaciones y sandboxes.
* Restricciones en velocidad para mensajes MQTT, MQTT Broker, eventos, webhooks y pulsaciones de botones virtuales.[[6]](#footnote-6)

### Limitaciones Comunes en Plataformas de Terceros

1. Restricciones en suscripciones:

Muchas plataformas de terceros ofrecen modelos de suscripción que limitan el acceso a características avanzadas y capacidades de almacenamiento, generando costos adicionales a medida que las necesidades de recolección de datos crecen.

1. Tipos de Licencia:

Las licencias pueden imponer restricciones en usuarios, dispositivos o datos, generando dificultades ante el crecimiento de actividades o una mayor participación de usuarios.

1. Dependencia de Infraestructura Externa:

La dependencia de la infraestructura y políticas de terceros puede resultar en interrupciones inesperadas en la recolección y gestión de datos.

1. Limitaciones en la Personalización:

La falta de flexibilidad en la personalización puede dificultar la adaptación a necesidades específicas del laboratorio.

1. Falta de Transparencia en Costos:

Ciertas plataformas pueden carecer de transparencia en cuanto a los costos de sus planes o los gastos vinculados al uso intensivo de recursos, lo que podría dar lugar a sorpresas financieras para los usuarios.

### Relevancia de las API en el Ámbito Tecnológico

En la presente era digital, las API se configuran como el componente conectivo esencial de nuestro entorno interconectado. La analogía pertinente sería considerar a una API como un puente que facilita la comunicación fluida entre distintos sistemas de software, permitiéndoles compartir información de manera eficiente. Este proceso posibilita la interacción sin inconvenientes entre el frontend, la interfaz de una aplicación o sitio web visible para el usuario, y el backend, compuesto por la base de datos y el servidor, a través de diversas aplicaciones, sitios web o servicios. Esta armonía digital constituye la base para un flujo eficaz de datos y acciones en el ámbito digital.

La vertiginosa evolución tecnológica ha modificado sustancialmente la manera en que nos relacionamos con el mundo digital, abriendo nuevas perspectivas tanto para empresas como para individuos. En el epicentro de esta revolución digital, emerge una herramienta fundamental que, aunque pasa desapercibida para el usuario promedio, resulta esencial para el funcionamiento de prácticamente todos los servicios o aplicaciones en línea: las interfaces de programación de aplicaciones, conocidas como API.

#### Como funciona una API

Una forma sencilla de comprender cómo funcionan es observar un ejemplo común: el procesamiento de pagos de terceros. Cuando un usuario compra un producto en un sitio de comercio electrónico, es posible que se le solicite "Pagar con PayPal" u otro tipo de sistema de terceros. Esta función se basa en API para realizar la conexión.

1. Cuando el comprador hace clic en el botón de pago, una API llama para recuperar información, también conocida como solicitud. Esta solicitud se procesa desde una aplicación al servidor web a través del Identificador uniforme de recursos (URI) de la API e incluye un verbo de solicitud, encabezados y, a veces, un cuerpo de solicitud.
2. Después de recibir una solicitud válida de la página web del producto, la API realiza una llamada al programa externo o servidor web, en este caso, el sistema de pago de terceros.
3. El servidor envía una respuesta a la API con la información solicitada.
4. La API transfiere los datos a la aplicación solicitante inicial, aquí el sitio web del producto.

Si bien la transferencia de datos variará según el servicio web que se utilice, todas las solicitudes y respuestas se realizan a través de una API. No hay visibilidad en la interfaz de usuario, lo que significa que las API intercambian datos dentro de la computadora o aplicación y aparecen ante el usuario como una conexión perfecta.

#### Beneficios de las API

Las API simplifican el diseño y desarrollo de nuevas aplicaciones y servicios, y la integración y gestión de los existentes. Pero ofrecen otros beneficios importantes a los desarrolladores y a las organizaciones en general [6].

* Mejora de la colaboración

La empresa media utiliza casi 1.200 [7] aplicaciones en la nube (el enlace reside fuera de ibm.com), muchas de las cuales están desconectadas. Las API permiten la integración para que estas plataformas y aplicaciones puedan comunicarse entre sí sin problemas. Gracias a esta integración, las empresas pueden automatizar los flujos de trabajo y mejorar la colaboración en el lugar de trabajo. Sin las API, muchas empresas carecerían de conectividad, lo que provocaría silos de información que comprometerían la productividad y el rendimiento.

#### Innovación acelerada

Las API ofrecen flexibilidad, lo que permite a las empresas establecer conexiones con nuevos socios comerciales, ofrecer nuevos servicios a su mercado existente y, en última instancia, acceder a nuevos mercados que pueden generar retornos masivos e impulsar la transformación digital.

#### Monetización de datos

Muchas empresas optan por ofrecer API de forma gratuita, al menos inicialmente, para poder crear una audiencia de desarrolladores en torno a su marca y forjar relaciones con posibles socios comerciales. Si la API otorga acceso a activos digitales valiosos, la empresa   
los monetiza vendiendo el acceso. Esto se conoce como economía API.

#### Sistema de seguridad

Las API separan la aplicación solicitante de la infraestructura del servicio que responde y ofrecen capas de seguridad entre las dos mientras se comunican. Por ejemplo, las llamadas API normalmente requieren credenciales de autenticación; Los encabezados HTTP, las cookies o las cadenas de consulta pueden proporcionar seguridad adicional durante el intercambio de datos y una API Gateway [8] puede controlar el acceso para minimizar aún más las amenazas a la seguridad.

#### Seguridad y privacidad del usuario final

Al igual que las API ofrecen una protección añadida dentro de una red, también pueden proporcionar otra capa de protección a los usuarios personales. Cuando un sitio web solicita la ubicación de un usuario, que se proporciona a través de una API de ubicación, el usuario puede decidir si permite o deniega esta solicitud. Muchos navegadores web y sistemas operativos móviles, como iOS, incorporan estructuras de permisos cuando las API solicitan acceso a las aplicaciones y sus datos. Cuando la aplicación debe acceder a archivos a través de una API, los sistemas de archivos como Windows, Mac y Linux utilizan permisos para ese acceso.

La importancia fundamental de las API en la era digital trasciende la mera conectividad, ofreciendo numerosas ventajas que transforman la forma en que interactuamos con la tecnología y aportan un valor inmenso tanto a los usuarios como a las empresas.

Las API permiten ahorrar tiempo, dinero y esfuerzo al reducir la necesidad de que los desarrolladores reinventen la rueda. Al aprovechar las API existentes, se evita la laboriosa tarea de construir cada componente desde cero. Esto posibilita que las empresas se enfoquen en crear funciones únicas y mejorar las experiencias de los usuarios. Además, impulsan la innovación al permitir la combinación de diferentes funcionalidades y fuentes de datos, dando lugar a la aparición de nuevas aplicaciones y servicios que de otra manera no serían posibles. La fusión tecnológica facilitada por las API abre la puerta a la personalización y adaptación a medida, otorgando a los usuarios un mayor control sobre sus experiencias digitales. Ya sea personalizando la interfaz de un smartphone o configurando sus aplicaciones favoritas, las API permiten a los usuarios dar forma a las experiencias digitales según sus preferencias, convirtiéndose así en herramientas indispensables en el mundo actual, hiper personalizado y centrado en el cliente.

## Herramientas de Recolección de Datos

En el desarrollo de LIITEC API, la recolección de datos es un aspecto fundamental para la adquisición de información proveniente de diversos sensores. En esta sección, se explorarán en detalle las herramientas de LIITEC para esta tarea, examinando los diferentes tipos de sensores sus características.

A través de la siguiente tabla, se presentará una visión completa de los sensores, brindando una comprensión exhaustiva de su funcionamiento y su contribución al proyecto.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sensor | Mide | Tipo | Unidad de medida | Descripción |
| **GYML8511 (\*)** | Intensidad UV | Analógico | mW/cm2 - Microwatts por centímetro cuadrado. | * Detecta luz con una longitud de onda entre 280-390nm, este rango cubre tanto al espectro UV-B como al UV-A. |
| **DHT22 (\*)** | Temperatura y Humedad | Digital | Grados Celsius “°C”, Fahrenheit “°F”, y Humedad Relativa “RH” | * + Proporciona mediciones de temperatura y humedad   + Rango de medición de temperatura: -40 °C a 80 °C.   + Precisión de la medición de temperatura: +/-0.5 °C.   + Rango de medición de humedad: 0% a 100% RH.   + Precisión de la medición de humedad: +/-2-5% dependiendo de la temperatura. |
| HW-08 | Humedad de suelo | Analógico | % Porcentaje de humedad. | * Sensor de humedad de suelo. * Entrega un valor de voltaje entre 0 y 5 V dependiendo de la cantidad de humedad detectada por el electrodo. |
| **MQ2** | GLP y GNV | Analógico | PPM - Partes por millón. | * Sensor de concentraciones de gas GLP y GNV en el aire. * Puede detectar concentraciones desde 300 hasta 10000 ppm. • Rango de detección: 300 a 10000 ppm. * Gas característico: 1000ppm, Isobutano. |
| **MQ3** | Alcohol en aire | Analógico | PPM - Partes por millón. | * Sensor detecta la concentración de alcohol en aire. * Detección de: Alcohol, Gasolina/Bencina y Etanol. * Especialmente sensible al: Alcohol. * Voltaje de Operación: 5V. * Corriente de Operación: 150mA. * Dispositivo base: Sensor MQ3. * Rango de detección: 0.05 a 10 mg/L. * Consumo de potencia calorífica: menos que 750mW. |
| **MQ135 (\*)** | NH3, NOx, alcohol, Benceno, Humo y CO2. | Digital y Analógico | PPM - Partes por millón. | * Sensor es capaz de detectar un amplio rango de gases que incluye: NH3, NOx, alcohol, Benceno, Humo y CO2. * 10 – 300 ppm Amoníaco (NH3) y Alcohol. * 10 – 1000 ppm Benceno. |
| PH-4502C | pH | Digital y Analógico | pH - Potencial de hidrógeno | * Sensor que mide el pH de una solución. * El pH se mide en una escala de 0 a 14, donde 7 es neutral, menores a 7 son ácidos y mayores a 7 son alcalinos. |
| LGZD-V1.1 | Turbiedad del agua | Digital y Analógico | NTU - Unidad nefelométrica de turbidez | * Sensor de turbiedad del agua. * Detectar partículas en suspensión en el agua mediante la medición de la transmitancia de luz y la frecuencia que cambia con la cantidad de sólidos suspendidos totales (SST) en el agua. |
| MF01 | Fuerza o Presión | Analógico | N - Newtons, g - gramos (N \* 100) | * Sensor de fuerza o presión. * Fuerza de actuación: 30g min. * Rango de sensibilidad: 30 a 10,000g. |
| YF-S201 | Flujo del agua | Analógico | L/min - Litros por minuto | * Sensor de flujo del agua. * Voltaje de operación: 5V - 18V DC. * Consumo de corriente: 15mA (5V). * Capacidad de carga: 10mA (5 VDC). * Salida: Onda cuadrada pulsante. * Rango de Flujo: 1-30L/min. |
| KY038 | Ruido | Digital y Analógico | dB - Decibeles | * Sensor de ruido. * Permite detectar un mínimo ruido producido en el ambiente. * Gama de frecuencias: 100 – 10.000 Hz. * Sensibilidad: – 46 ± 2,0, (0dB = 1V / Pa) a 1K Hz. * La sensibilidad mínima a ruido: 58 dB. |

Tabla 1 - Información general sobre los sensores de LIITEC

Aunque esta lista abarca la mayoría de los sensores utilizados por LIITEC para medir el entorno, en la implementación de la API se considerarán únicamente aquellos sensores que LIITEC está actualmente utilizando en campo:

* **GYML8511:** Este sensor registra la intensidad de los rayos ultravioleta.
* **DHT22:** Se encarga de medir la temperatura y la humedad ambiental.
* **MQ135:** Este sensor evalúa la calidad del aire en el entorno.

## Arquitectura y Tecnologías Utilizadas

En este capítulo, se explorará en detalle la arquitectura subyacente y las herramientas tecnológicas que han sido fundamentales en el desarrollo de la API. La elección de una arquitectura sólida y la selección cuidadosa de herramientas desempeñan un papel crucial en el éxito y la eficiencia del sistema. A lo largo de este capítulo, se analizará la estructura general de la API, destacando los componentes esenciales y el modelo de datos subyacente. Además, se proporcionará una visión detallada de las herramientas tecnológicas utilizadas, desde el lenguaje de programación y la base de datos hasta los frameworks y estándares de seguridad implementados. Este análisis integral sentará las bases para comprender la solidez técnica y la capacidad adaptativa de la API en el contexto específico del Laboratorio LIITEC.

### Arquitectura

Al abordar la gestión de datos, surge un desafío significativo: cómo facilitar la transferencia eficiente de información entre distintos puntos. Este desafío se vuelve aún más crucial al trabajar con sensores, ya que se anticipa una generación sustancial de datos, dada la permanencia y constante actividad de estos dispositivos en su entorno laboral. Para afrontar esta complejidad y lograr una gestión óptima de la información recopilada, es esencial contar con una arquitectura sólida que ordene y facilite el tratamiento de estos datos.

En este contexto, una estrategia que cobra gran relevancia es el proceso de extracción, transformación y carga (ETL), la cual se focaliza en extraer datos desde diversas fuentes, transformarlos según las necesidades específicas y cargarlos en un sistema de almacenamiento centralizado. Esta metodología no solo optimiza la eficiencia del proceso, sino que también posibilita una gestión más efectiva de grandes volúmenes de datos generados por sensores en su ambiente de trabajo continuo.

La implementación de una arquitectura ETL en este proyecto no solo resuelve la problemática del flujo de datos, sino que también abre la puerta a una gestión más estructurada y eficaz de la información generada por los sensores.

#### Justificación de la Elección de ETL

La elección de implementar el proceso de ETL en el contexto de este proyecto se fundamenta en la necesidad imperante de gestionar eficientemente la ingente cantidad de datos generados por los sensores. Al adoptar la metodología ETL, se busca abordar específicamente los desafíos inherentes a la recolección, transformación y almacenamiento de esta abundante información, proporcionando una serie de beneficios clave:

* Eficiencia

Automatizar la extracción, transformación y carga de datos mejora significativamente el proceso, ahorrando tiempo y recursos esenciales al manejar grandes cantidades de datos de sensores de manera eficiente.

* Escalabilidad

Manejar volúmenes crecientes de datos de manera eficaz es esencial para adaptarse a la continua generación de datos por parte de los sensores en el laboratorio, asegurando que la solución sea flexible y sostenible a lo largo del tiempo.

* Flexibilidad

Extraer datos de diversas fuentes y cargarlos en sistemas diferentes facilita la integración efectiva de información variada. Esto es especialmente valioso al trabajar con sensores, permitiendo recopilar datos de múltiples fuentes para análisis integrales.

* Automatización

Programar y ejecutar automáticamente procesos ETL ahorra tiempo y recursos, permitiendo centrarse en análisis más significativos. La automatización es clave, especialmente en entornos donde la actualización de datos es constante.

* Mejora de la Calidad de Datos

Validar, limpiar y estandarizar datos durante la transformación contribuye significativamente a mejorar la calidad de los datos generados por sensores. Esta mejora es esencial para garantizar la precisión y confiabilidad requeridas en investigaciones educativas.

#### Fases del proceso ETL

En esta sección, exploraremos con mayor profundidad las distintas fases del proceso ETL (Ilustración 1) que conforman la columna vertebral de nuestra arquitectura para la gestión de datos provenientes de sensores.

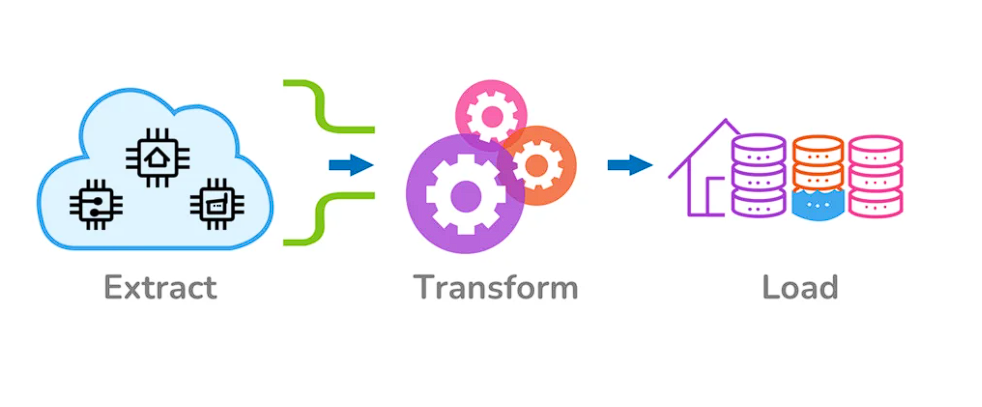


Ilustración 1- Representación gráfica del proceso ETL

1. Proceso de Extracción de datos

En el inicio, extraemos datos recopilando información en bruto de varias fuentes, como bases de datos, archivos planos o dispositivos IOT. Estos datos pueden tener diferentes estructuras y provenir de diversas fuentes, como sistemas ERP o CRM, sitios web, tiendas de comercio electrónico, soluciones de marketing o hojas de cálculo Excel. La extracción de datos es esencial en ETL, ya que proporciona la materia prima y la información crítica para las etapas siguientes del proceso.

1. Proceso de Transformación de datos–

Después de extraer los datos, el siguiente paso es la transformación. Esto es crucial, ya que asegura que todos los datos se limpien y formateen de manera adecuada para el sistema de destino y su estructura de base de datos. La transformación adapta dinámicamente los datos extraídos al formato requerido, realizando acciones como eliminar duplicados, normalizar formatos, agregar información o aplicar cálculos específicos. Este proceso garantiza que los datos sean consistentes, precisos y libres de errores antes de cargarlos en la base de datos final del sistema empresarial.

1. Proceso de Carga de datos

El último paso es la carga de datos, donde los datos transformados se introducen en el sistema empresarial o la base de datos final. Esto puede ser un almacén de datos, un lago de datos o incluso una sencilla hoja de cálculo de Excel. Durante la carga, se estructuran los datos para facilitar su acceso, consulta y análisis. La base de datos de destino se optimiza para diferentes tipos de datos y usos finales, lo que puede implicar la creación de modelos específicos, particiones de datos o la implementación de índices.

#### Arquitectura IOT junto a proceso ETL

ETL pipeline para un sistema IOT

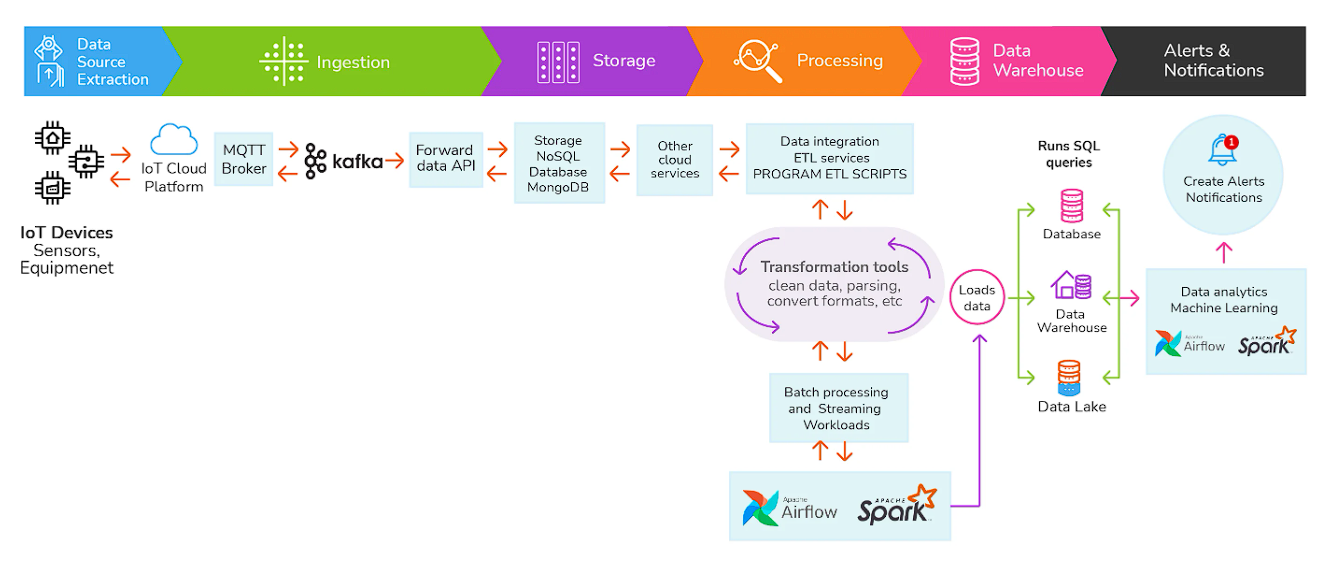


Ilustración 2 - Proceso ETL en sistema IOT

La estructura del proyecto toma inspiración de la publicación "*Build a Real-time ETL Pipeline for an IoT System*" de Jose Luis Amoros - abril 20, 2022 [9]. El artículo aborda la creación de un pipeline para extraer datos de sensores IoT, transformarlos en información útil y cargarlos en almacenes de datos o data warehouses.

La esencia de esta arquitectura reside en la gestión eficiente de datos provenientes de dispositivos IoT, los cuales generan información a alta velocidad desde diversas fuentes. La implementación de procesamiento en tiempo real destaca, permitiendo el análisis eficiente de grandes volúmenes de datos para tomar decisiones empresariales inmediatas.

En términos de arquitectura, se propone un flujo de trabajo que comienza recolectando datos de dispositivos IoT, sensores y equipos. Estos datos luego los ingiere un almacenamiento en la nube para permitir un acceso rápido en tiempo real, facilitando su procesamiento a través de soluciones como ETL y análisis. Posteriormente, los datos se pueden enviar a almacenes de datos o directamente a otras aplicaciones, como dashboards, para activar alertas, notificaciones, correos electrónicos y más.

Este enfoque integral aborda la complejidad de la ingesta de datos, procesamiento y almacenamiento, garantizando la coherencia y eficiencia en el flujo de datos antes de su uso en aplicaciones empresariales y productos computacionales. La adaptabilidad de esta arquitectura permite la implementación de enfoques más complejos según las necesidades específicas del proyecto, asegurando la optimización del procesamiento de datos provenientes de dispositivos IoT en tiempo real.

Flujo de los datos

La (Ilustración 2) muestra un pipeline de flujo de datos, que es una secuencia de pasos utilizada para mover y procesar datos desde su origen hasta un destino donde se pueden analizar y utilizar.

1. Extracción de la Fuente de Datos:

Los datos son generados por dispositivos IoT como pueden ser sensores y equipos. Estos dispositivos están conectados a una Plataforma Cloud IoT.

1. Ingesta

Los datos de los dispositivos IoT se envían a un Broker MQTT para luego ser publicados en Apache Kafka, que es una plataforma de streaming distribuida que puede manejar grandes volúmenes de datos y permite que los datos se reenvíen a través de una API de datos.

1. Almacenamiento

Una vez que los datos han sido ingeridos, se almacenan en una base de datos NoSQL, en este caso MongoDB, que está diseñado para manejar grandes volúmenes de datos y proporciona un alto rendimiento para las operaciones de lectura y escritura. Los datos también pueden ser almacenados en otros servicios en la nube según sea necesario.

1. Procesamiento

Los datos almacenados atraviesan un proceso de tratamiento, que incluye:

* + Integración de datos a través de servicios y scripts ETL, que consolidan datos de diferentes fuentes.
  + Se utilizan herramientas de transformación para limpiar los datos, analizarlos y convertirlos en los formatos requeridos.
  + Se manejan cargas de trabajo de procesamiento por lotes y en streaming, lo que significa que los datos pueden ser procesados en grandes lotes o en flujos en tiempo real.

1. Almacén de Datos

Los datos procesados se cargan luego en un almacén de datos y/o un lago de datos. Un almacén de datos es un repositorio centralizado para datos estructurados, optimizado para consultas y análisis. Un lago de datos, por otro lado, puede almacenar datos no estructurados y semi-estructurados a gran escala.

1. Alertas y Notificaciones

Los datos dentro del almacén de datos pueden ser utilizados para ejecutar consultas SQL para un análisis más profundo. Basándose en el análisis, se pueden realizar tareas de análisis de datos y aprendizaje automático utilizando herramientas como Apache Airflow (para orquestar flujos de trabajo) y Apache Spark (para el procesamiento de datos a gran escala). Los conocimientos obtenidos de estos análisis pueden desencadenar la creación de alertas y notificaciones para informar a los interesados o desencadenar procesos automatizados.

## Tecnologías Empleadas

En el desarrollo de cualquier proyecto tecnológico, la selección adecuada de las herramientas y tecnologías desempeña un papel crítico en el logro de los objetivos establecidos. En el contexto de la creación de la LIITEC API, se ha dado especial atención a las Tecnologías Empleadas. Este apartado explora las plataformas, lenguajes de programación, frameworks y demás herramientas que han sido cuidadosamente elegidas para dar forma a la infraestructura tecnológica de la API. El análisis detallado de estas tecnologías proporcionará una visión clara de la base sobre la cual se construye la solución propuesta

### API web REST

La elección de una API web RESTful para LIITEC se fundamenta en la organización eficiente de sus recursos y la facilidad de manejo por parte de los usuarios. La estructura intuitiva y la disposición lógica de los recursos en este tipo de API permiten una interacción más natural y amigable, optimizando la experiencia del usuario y simplificando la gestión interna de los servicios.

#### Ventajas de las APIs REST:

La adopción de APIs REST en lugar de protocolos estándar, como XML-RPC o SOAP, se basa en varios factores. En comparación con SOAP, que utiliza un intercambio de datos XML y una fuerte unión entre los componentes, las APIs REST presentan una arquitectura más desacoplada, lo que facilita la implementación y adaptabilidad a las necesidades de LIITEC. Entre las ventajas específicas se destacan.

1. Escalabilidad

La separación entre el cliente y el servidor permite una escalabilidad eficiente, gestionando fácilmente el crecimiento del producto con un equipo de desarrollo sin mayores complicaciones.

1. Flexibilidad y Portabilidad

La correcta transmisión de datos en cada petición posibilita la migración entre servidores y cambios en la base de datos sin restricciones. La flexibilidad permite alojar el frontend y el backend en servidores diferentes, simplificando la administración del sistema.

1. Independencia

La clara separación entre el cliente y el servidor facilita desarrollos independientes en diferentes partes del proyecto. Además, la API REST se adapta a diversas sintaxis y plataformas, ofreciendo flexibilidad en entornos de desarrollo.

1. Seguridad

Implementación de medidas de seguridad como autenticación y autorización mediante estándares como OAuth, asegurando la idoneidad de las APIs REST para aplicaciones sensibles.

1. Documentación

Las APIs REST suelen contar con documentación clara y accesible, describiendo de manera completa los recursos disponibles, sus URI y los métodos HTTP utilizables. Esto facilita la comprensión y el uso efectivo de la API en el contexto de LIITEC.

#### Diseño de API en torno a los recursos

Según Microsoft [10] se centran en las entidades empresariales que expone la API web. Por ejemplo, en un sistema de comercio electrónico, las entidades principales podrían ser clientes y pedidos. La creación de un pedido se puede lograr mediante el envío de una solicitud HTTP POST que contiene la información del pedido. La respuesta HTTP indica si el pedido se realizó correctamente o no. Siempre que sea posible, los URI de recursos deben basarse en nombres (el recurso) y no en verbos (las operaciones en el recurso).

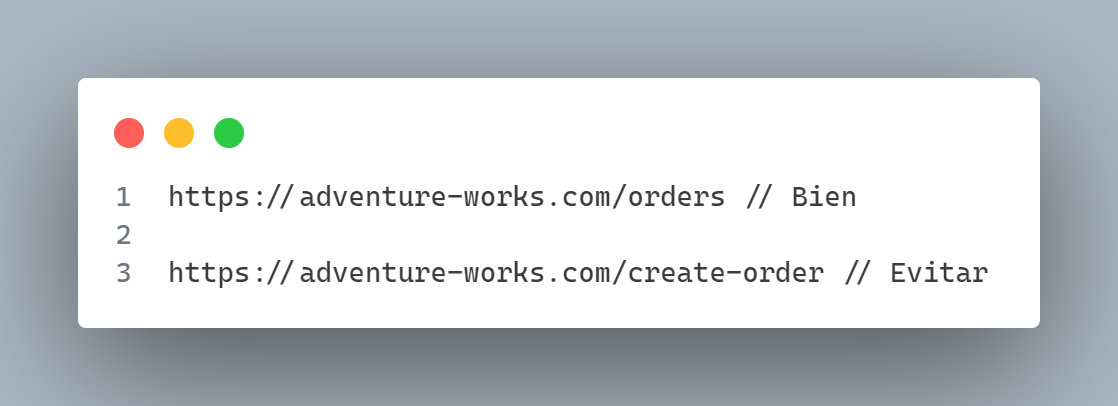


Ilustración 3 - Recomendación de escritura de verbos en URLs

Un recurso no tiene que estar basado en un solo elemento de datos físico. Por ejemplo, un recurso de pedido podría implementarse internamente en forma de varias tablas de una base de datos relacional, pero presentarse al cliente como una única entidad. Evite la creación de API que simplemente reflejen la estructura interna de una base de datos. La finalidad de REST es modelar entidades y las operaciones que una aplicación puede realizar sobre esas entidades. Un cliente no debe exponerse a la implementación interna.

##### Flujo de diseño

Según Google [11] se sugiere seguir los pasos a continuación durante el diseño de API orientadas a recursos:

* Determinar los tipos de recursos que proporciona una API.
* Determinar la relación entre recursos.
* Decidir los esquemas de nombres de recurso según los tipos y las relaciones.
* Decidir los esquemas de recursos.
* Adjuntar un conjunto de métodos mínimo a los recursos.

##### Recursos

Por lo general, una API orientada a recursos se modela como una jerarquía de recursos en la que cada nodo es un “*recurso simple”* o un “*recurso de colección”*. Por conveniencia, se los suele llamar recurso y colección, respectivamente.

* Una colección contiene una lista de recursos del mismo tipo. Por ejemplo, un usuario tiene una colección de contactos.
* Un recurso tiene un estado y cero o más subrecursos. Cada subrecurso puede ser siempre o de colección.

Por ejemplo, una API de Gmail tiene una colección de usuarios, cada usuario tiene una colección de mensajes, una de conversaciones, una de etiquetas, un recurso de perfil y muchos recursos de opciones de configuración.

Si bien hay cierta alineación conceptual entre los sistemas de almacenamiento y las API de REST, un servicio con una API orientada a recursos no es necesariamente una base de datos y es muy flexible respecto a su forma de interpretar métodos y recursos. Por ejemplo, con la creación de un evento de calendario (recurso) se pueden crear eventos adicionales para los asistentes, enviar invitaciones por correo electrónico a estos, reservar salas de conferencia y actualizar programas de videoconferencia.

#### Definición de las operaciones de API en términos de métodos HTTP

El protocolo HTTP define una serie de métodos que asignan significado semántico a una solicitud. Los métodos HTTP comunes que usan la mayoría de las API web RESTful son:

* **GET** recupera una representación del recurso en el URI especificado. El cuerpo del mensaje de respuesta contiene los detalles del recurso solicitado.
* **POST** crea un nuevo recurso en el URI especificado. El cuerpo del mensaje de solicitud proporciona los detalles del nuevo recurso. Tenga en cuenta que POST también puede usarse para desencadenar operaciones que en realidad no crean recursos.
* **PUT** crea o sustituye el recurso en el URI especificado. El cuerpo del mensaje de solicitud especifica el recurso que se va a crear o actualizar.
* **PATCH** realiza una actualización parcial de un recurso. El cuerpo de la solicitud especifica el conjunto de cambios que se aplican al recurso.
* **DELETE** quita el recurso en el URI especificado.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Recurso | POST | GET | PUT | DELETE |
| /customers | Crear un nuevo cliente | Recuperar todos los clientes | Actualización masiva de clientes | Eliminar todos los clientes |
| /customers/1 | Error | Recuperar los detalles del cliente 1 | Actualizar los detalles del cliente 1 si existe | Quitar al cliente 1 |
| /customers/1/orders | Crear un nuevo pedido para el cliente 1 | Recuperar todos los pedidos del cliente 1 | Actualización masiva de pedidos del cliente 1 | Recuperar todos los pedidos del cliente 1 |

Tabla 2 - Uso correcto de las operaciones en una API REST

#### Autenticación y validación en API REST

##### Métodos de autenticación

Como se aprecia en la Ilustración 4, básicamente tenemos dos formas de implementar la autenticación en servidor [12]:

1. **Basada en cookies**, la más utilizada:

* El servidor guarda la cookie para autenticar al usuario en cada request.
* Habrá que tener un almacen de sesiones: en base de datos, Redis, etc.

1. Basada en tokens, se confía en un token firmado que se envía al servidor en cada petición.

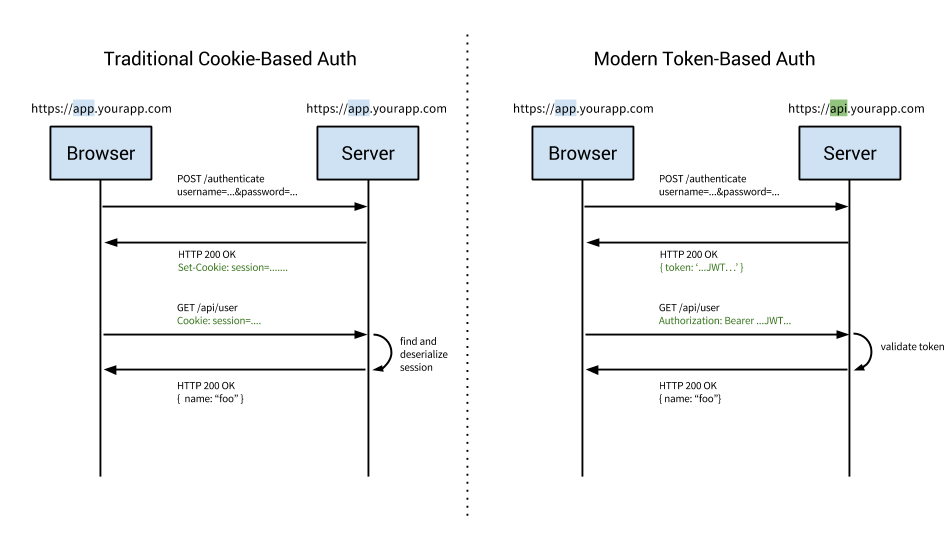


Ilustración 4 - Pipeline Metodos de Autenticación en APIs

##### ¿Qué es un token?

* Un token es un valor que nos autentica en el servidor
  + Normalmente se consigue después de hacer login mediante usuario/contraseña.
* ¿Cómo se genera el token?
  + Normalmente un hash calculado con algún dato (p.ej. login del usuario + clave secreta).
  + Además, el token puede llevar datos adicionales como el login
* ¿Cómo comprueba el servidor que es válido?
  + Generando de nuevo el Hash y comprobando si es igual que el que envía el usuario (100% stateless[[7]](#footnote-7)).
  + O bien habiendo almacenado el Hash en una base de datos asociado al usuario y simplemente comprobando que coincide.

Beneficios de usar tokens

* Uso entre dominios
  + Cookies + CORS no se llevan bien.

### Docker

Docker [13] es una plataforma de código abierto que permite a los desarrolladores construir, desplegar, ejecutar, actualizar y gestionar contenedores. Los contenedores son componentes ejecutables estandarizados que combinan el código fuente de una aplicación con las bibliotecas y dependencias del sistema operativo necesarias para ejecutar ese código en cualquier entorno. Simplifican el desarrollo y la entrega de aplicaciones distribuidas y se han vuelto cada vez más

populares a medida que las organizaciones se desplazan hacia el desarrollo cloud-native[[8]](#footnote-8) y los entornos híbridos multicloud[[9]](#footnote-9). Aunque es posible crear contenedores sin Docker, trabajando directamente con las capacidades incorporadas en Linux y otros sistemas operativos, Docker hace que la contenerización sea más rápida, fácil y segura.

#### Orquestación de Docker

En el ámbito de proyectos extensos o escalables, la gestión simultánea de múltiples contenedores se convierte en un aspecto crítico. En este contexto, destaca la utilidad de herramientas específicas diseñadas para abordar este desafío, siendo Docker Compose una destacada elección.

Docker Compose

Es una herramienta versátil que permite a los desarrolladores gestionar aplicaciones con múltiples contenedores en un mismo host de Docker. Facilita la creación de archivos YAML para definir los servicios incluidos en la aplicación, lo que posibilita el despliegue y ejecución de contenedores con una sola orden. La sintaxis YAML, al ser independiente del lenguaje, se adapta a diversos programas escritos en Java, Python, Ruby y otros. Además, Docker Compose posibilita la definición de volúmenes persistentes, la especificación de nodos base, y la documentación y configuración de dependencias de servicios.

Un ejemplo práctico de la utilidad de Docker Compose sería en la creación de un proyecto que involucre NodeJS y MongoDB. Al emplear esta herramienta, se puede crear una única imagen que inicie ambos contenedores como un servicio, evitando la necesidad de iniciar cada uno de manera individual.

#### ¿Por qué Docker?

El uso de Docker en el desarrollo de LIITEC API ofrece ventajas significativas en términos de consistencia y eficiencia en la gestión de versiones en diversos entornos y plataformas.

Imaginemos un escenario donde un grupo de desarrolladores trabaja en la versión 5.0 de MongoDB. Si un nuevo miembro se une al equipo, tradicionalmente se vería enfrentado a la tarea de configurar manualmente la misma versión de MongoDB con parámetros exactos, lo cual podría propiciar conflictos si otros desarrolladores ya están utilizando una versión diferente.

Para evitar estos desafíos, Docker proporciona una solución eficiente al permitir la creación de una imagen personalizada de MongoDB con una configuración específica, que puede ser almacenada internamente en Docker Hub. Cuando un nuevo miembro se incorpora al equipo, puede extraer fácilmente esta imagen y desplegar la implementación sin necesidad de configuraciones manuales complicadas.

Este enfoque ofrece ventajas significativas:

* Coherencia en el Equipo: Todos los miembros del equipo pueden garantizar el uso de las mismas versiones y configuraciones sin depender de recursos externos.
* Facilidad de Configuración: La implementación en Docker simplifica el proceso de configuración, permitiendo iniciar y detener el servidor de manera rápida y sencilla a través de Docker Desktop.

### Broker MQTT – Eclipse Mosquitto

Según varias fuentes [14] Eclipse Mosquitto es un broker de mensajes de código abierto bajo la licencia EPL/EDL, que implementa las versiones 5.0, 3.1.1 y 3.1 del protocolo MQTT (Message Queue Telemetry Transport). Destaca por su eficiencia y soporte para dispositivos desde computadoras de placa única de bajo consumo hasta servidores completos. Su capacidad para establecer puentes con otros servidores MQTT, incluyendo instancias de Mosquitto, amplía su versatilidad. Además, Mosquitto es reconocido por su ligereza, siendo ideal para aplicaciones de máquina a máquina (M2M) o Internet de las cosas (IoT), donde el ancho de banda es limitado. Su implementación eficaz y su adaptabilidad lo posicionan como una elección sólida para la comunicación a través del protocolo MQTT en una variedad de entornos y dispositivos.

#### Protocolo MQTT

Según Amazon [15] MQTT es un protocolo de mensajería basado en estándares, o un conjunto de reglas, que se utiliza para la comunicación de un equipo a otro. Los sensores inteligentes, los dispositivos portátiles y otros dispositivos de Internet de las cosas (IoT) generalmente tienen que transmitir y recibir datos a través de una red con recursos restringidos y un ancho de banda limitado. Estos dispositivos IoT utilizan MQTT para la transmisión de datos, ya que resulta fácil de implementar y puede comunicar datos IoT de manera eficiente. MQTT admite la mensajería entre dispositivos a la nube y la nube al dispositivo.

##### Componentes MQTT

MQTT implementa el modelo de publicación o suscripción mediante la definición de clientes y agentes, tal y como se muestra a continuación.

Cliente MQTT

Un cliente MQTT es cualquier dispositivo, desde un servidor hasta un microcontrolador, que ejecuta una biblioteca MQTT. Si el cliente envía mensajes, actúa como editor, y si recibe mensajes, actúa como receptor. Básicamente, cualquier dispositivo que se comunique mediante MQTT a través de una red puede denominarse dispositivo cliente MQTT.

Agente MQTT

El agente MQTT es el sistema de *backend* que coordina los mensajes entre los diferentes clientes. Las responsabilidades del agente incluyen recibir y filtrar mensajes, identificar a los clientes suscritos a cada mensaje y enviarles los mensajes. También se encarga de otras tareas como:

* La autorización y autenticación de clientes MQTT
* Pasar mensajes a otros sistemas para su posterior análisis
* El control de mensajes perdidos y sesiones de clientes

Conexión MQTT

Los clientes y los agentes comienzan a comunicarse mediante una conexión MQTT. Los clientes inician la conexión al enviar un mensaje *CONECTAR* al agente MQTT. El agente confirma que se ha establecido una conexión al responder con un mensaje *CONNACK*. Tanto el cliente MQTT como el agente requieren una pila TCP o IP para comunicarse. Los clientes nunca se conectan entre sí, solo con el agente.

##### Cómo funciona MQTT

A continuación, se proporciona una descripción general del funcionamiento de MQTT.

1. Un cliente MQTT establecer una conexión con el agente MQTT.
2. Una vez conectado, el cliente puede publicar mensajes, suscribirse a mensajes específicos o hacer ambas cosas.
3. Cuando el agente MQTT recibe un mensaje, lo reenvía a los suscriptores que están interesados.

Tema (tópico) de MQTT

El término “tema” se refiere a las palabras clave que utiliza el agente MQTT a fin de filtrar mensajes para los clientes de MQTT. Los temas están organizados jerárquicamente, de forma similar a un directorio de archivos o carpetas. Por ejemplo, considere un sistema doméstico inteligente que opera en una casa de varios pisos que tiene diferentes dispositivos inteligentes en cada uno de ellos. En ese caso, es posible que el agente MQTT organice temas como:

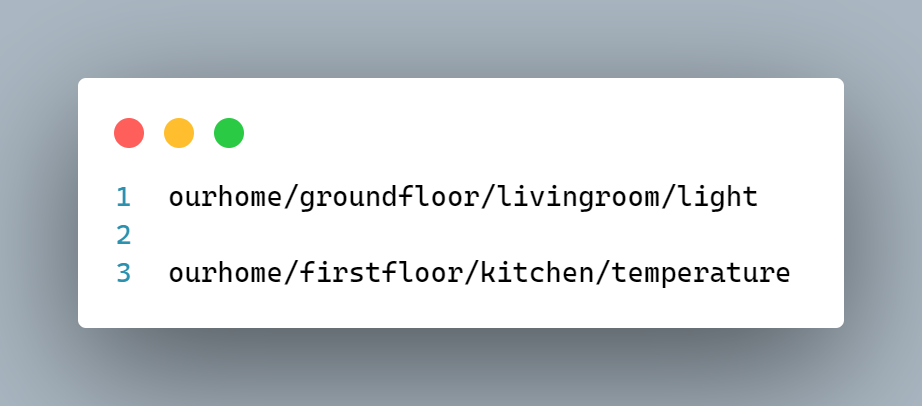


Ilustración 5 - Ejemplo de Tema (tópico) MQTT

**Publicación MQTT**

Los clientes MQTT publican mensajes que contienen el tema y los datos en formato de bytes. El cliente determina el formato de los datos, como datos de texto, datos binarios, archivos XML o JSON. Por ejemplo, es posible que una lámpara del sistema doméstico inteligente publique un mensaje sobre el tema “*salón”* o *“luz”*.

Suscripción MQTT

Los clientes MQTT envían un mensaje *SUBSCRIBE* (SUBSCRIBIRSE) al agente MQTT para recibir mensajes sobre temas de interés. Este mensaje contiene un identificador único y una lista de suscripciones. Por ejemplo, la aplicación de hogar inteligente en su teléfono quiere mostrar cuántas luces están encendidas en casa. Se suscribirá a la *luz* del tema y aumentará el contador para todos los mensajes *activados*.

##### Niveles de calidad de servicio [16]

MQTT facilita la entrega de mensajes de aplicación mediante niveles de servicio de calidad (QoS) definidos. Es importante destacar que el nivel de QoS utilizado para entregar un mensaje saliente puede diferir del utilizado para un mensaje entrante.

* QoS = 0 - Como máximo una vez

El mensaje se entrega como máximo una vez, si no, no se entrega. No se efectúa acuse de recibo la entrega del mensaje por la red. El mensaje no se almacena. El mensaje puede perderse si se desconecta el cliente o si falla el servidor. QoS=0 es la modalidad de transferencia más rápida. Se denomina a veces "transmitir y olvidar".

* QoS = 1 – Al menos una vez

El mensaje siempre se entrega, como mínimo, una vez. Si el emisor no recibe un acuse de recibo, el mensaje se envía de nuevo con el distintivo DUP establecido hasta que se reciba un acuse de recibo. Como resultado, un mismo mensaje se puede enviar varias veces al receptor y ser procesado varias veces.

* QoS = 2 - Exactamente una vez

Es la modalidad de transferencia más segura, pero la más lenta. Deben realizarse como mínimo dos pares de transmisiones entre el emisor y el receptor antes de que el mensaje pueda suprimirse de la parte del emisor. El mensaje puede procesarse en el receptor tras la primera transmisión.

##### Mosquitto Go Auth

Mosquitto, en su forma básica, es un broker accesible mediante la simple dirección del broker. Sin embargo, para reforzar la seguridad del broker, Mosquitto ofrece el "plugin Dynamic Security" [17], una extensión que proporciona funciones de autenticación y control de acceso actualizables en tiempo real mientras el broker está en funcionamiento. En este caso, se ha optado por implementar Mosquitto Go Auth [18], un complemento escrito principalmente en Go para brindar autenticación y autorización en el conocido broker MQTT de código abierto, Mosquitto. Este plugin, al estar mayormente desarrollado en Go, expone las funciones esenciales del complemento de autenticación de Mosquitto, adaptándose fácilmente a diversos tipos de back-end. De esta manera, Mosquitto Go Auth agrega una capa de seguridad y control sobre los usuarios que acceden al broker, mejorando su gestión y protegiendo la integridad del sistema.

### Express

Express.js, también conocido simplemente como "Express" [19], destaca como un marco de backend rápido y minimalista para Node.js. Este marco proporciona herramientas y características robustas para el desarrollo de aplicaciones de backend escalables. Con un sistema de enrutamiento eficiente y funciones simplificadas, Express permite ampliar el marco mediante la creación de componentes poderosos adaptados a diversos casos de uso. Además, ofrece un conjunto completo de herramientas para el manejo de solicitudes y respuestas HTTP, enrutamiento y middleware, facilitando la construcción de aplicaciones empresariales a gran escala. Su popularidad se refleja en ser la base de otros marcos web Node y en su capacidad para integrarse con motores de renderización de vistas. Utilizando el principio "Don't Repeat Yourself" (DRY), Express fomenta prácticas de desarrollo eficientes y una gestión coherente de aplicaciones web. Con una interfaz de línea de comandos (CLI) llamada Node Package Manager (NPM), Express agiliza el proceso de desarrollo y se mantiene como una elección destacada para desarrolladores que buscan un rendimiento ágil y confiable.

#### ¿Por qué Express?

En el proceso de selección de las tecnologías para desarrollar LIITEC API, Express.js emerge como una opción integral y eficiente. Esta elección se fundamenta en la experiencia previa [20] y las notables ventajas que ofrece, desde su capacidad para gestionar solicitudes y respuestas de manera ágil hasta su probada escalabilidad y robusto sistema de enrutamiento. A continuación, se exploran detalladamente algunas de las razones clave que respaldan la elección de Express.js para nuestro proyecto.

Escalabilidad

Express ha consolidado su reputación como un marco altamente escalable, respaldado por su amplia adopción en grandes empresas [21]. Su eficiente manejo de solicitudes y respuestas, incluso en proyectos web a gran escala, lo convierte en una opción ideal para LIITEC API. La presencia de módulos y paquetes adicionales de calidad contribuye a la creación de aplicaciones web fiables y preparadas para el crecimiento.

Apoyo de la Comunidad

La comunidad activa y comprometida respalda a Express.js, convirtiéndolo en el marco más popular para el backend de Node.js. Esta sólida base de apoyo ofrece a los desarrolladores recursos, paquetes y soluciones para abordar cualquier desafío de desarrollo. Además, la naturaleza de código abierto de Express.js fomenta la colaboración y la creación de extensiones, beneficiando no solo a los desarrolladores individuales, sino a toda la comunidad.

Potente Sistema de Enrutamiento

Express también destaca por su robusto sistema de enrutamiento incorporado, esencial para gestionar las solicitudes de manera eficiente. Este sistema permite organizar la estructura de la aplicación de forma clara y modular, facilitando la gestión de rutas y la creación de códigos más mantenibles. La capacidad de dividir el sistema de enrutamiento en archivos gestionables agiliza el desarrollo y evita la redundancia.

Middleware

Ofrece un conjunto de middleware que agiliza y personaliza el proceso de desarrollo. Los middlewares permiten ejecutar scripts antes o después de que una solicitud del cliente llegue al manejador de rutas, brindando flexibilidad y control sobre el flujo de la aplicación.

Esta funcionalidad es esencial para la introducción de scripts personalizados, como la verificación del estado de conexión de un usuario, la autenticación, etc. Optimizando así la experiencia del usuario.

### MongoDB

Para fundamentar la elección de MongoDB como base de datos documental, nos apoyaremos en el siguiente caso de uso narrado por Agustín Peláez - 24 de abril 2014, cofundador de Ubidots [22], que describe una aplicación de gestión del agua.

Supongamos que queremos medir los niveles de agua en un gran número de pozos. Una arquitectura de datos simplificada para esta aplicación tendría el siguiente aspecto:

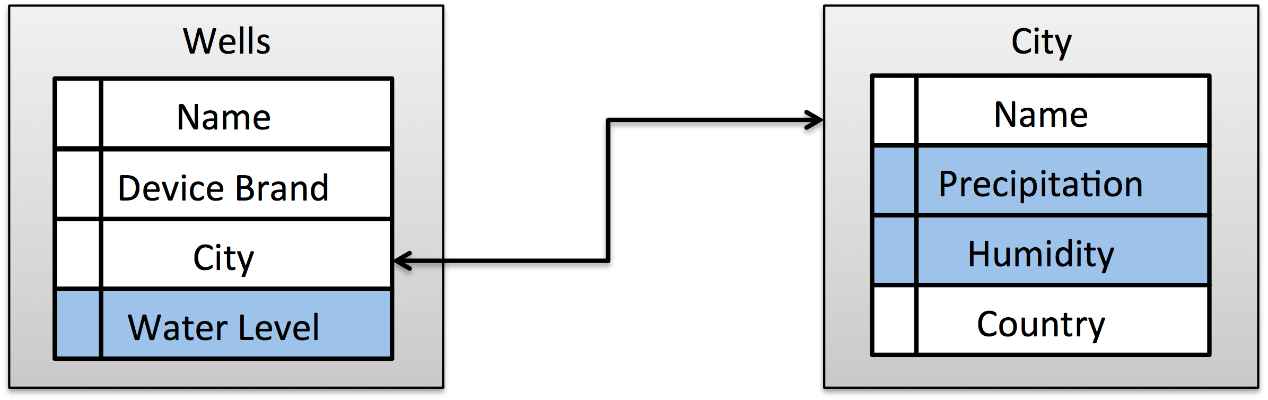


Ilustración 6 - Arquitectura simple para medir el nivel de agua en un pozo

Esto se ve bien y debería funcionar perfectamente usando una base de datos relacional. Pero entonces, 2 años después de que el sistema haya estado en funcionamiento, alguien tiene una idea:

*"Oye, ahora que compramos estos nuevos generadores diésel con acceso a Internet para alimentar las bombas de agua, ¡veamos sus datos en vivo!".*

Para realizar este cambio, tendríamos que añadir una nueva tabla llamada "Centrales eléctricas" y una nueva columna a la tabla "Pozos", marcada en rojo a continuación:

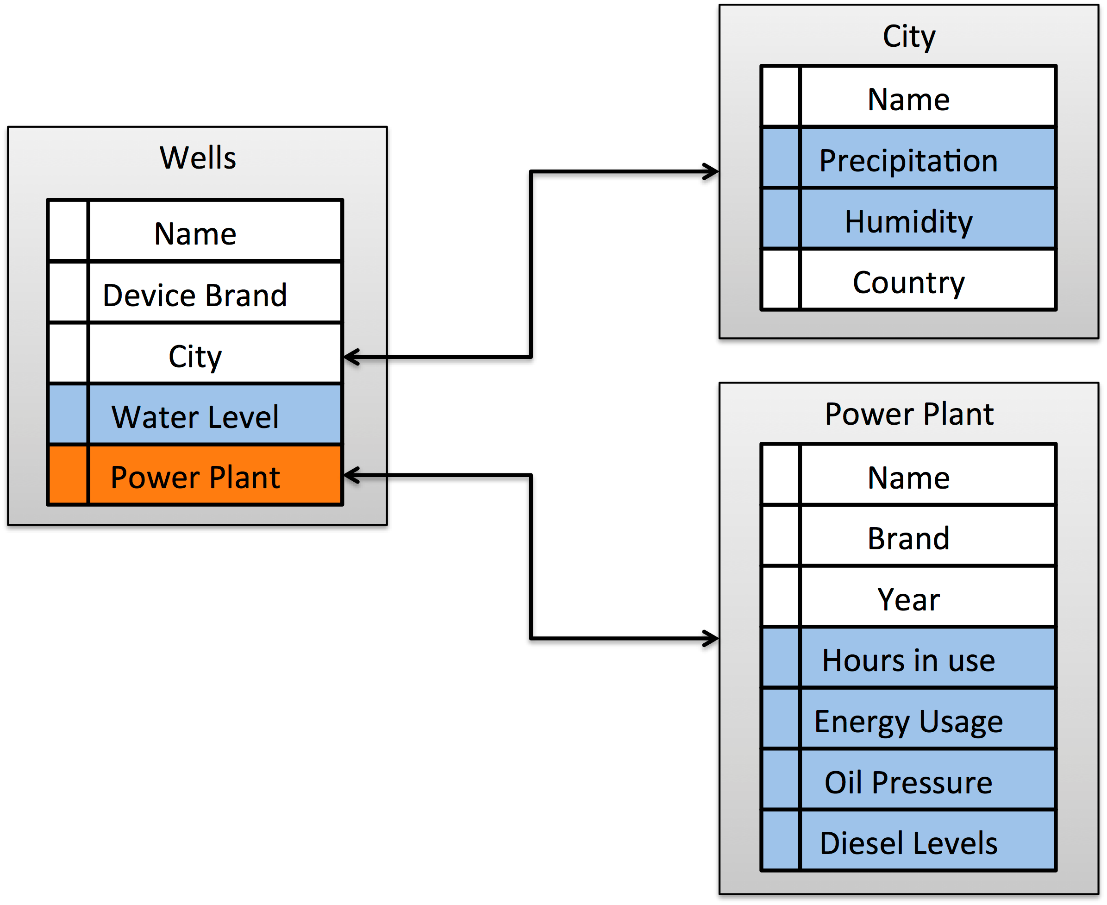


Ilustración 7 - Arquitectura para medir el agua de un pozo más generadores diésel

Pero en una base de datos relacional, este cambio requeriría que volvamos a escribir todas las líneas de la tabla para que incluyan un valor en la nueva columna. Con dos años de datos históricos, esto podría llevar un par de horas, bloqueando la base de datos a medida que sucede.

¿Te imaginas este procedimiento con cada cambio futuro? ¿Cómo afecta a la estabilidad, el tiempo de actividad y los requisitos del personal técnico del sistema?

Una forma interesante de manejar los datos de IoT es el enfoque orientado a documentos. En lugar de tablas, columnas y filas fijas, tiene documentos que describen cada objeto. Puedes pensar

en ello como miles de papeles (documentos) tirados por el suelo, en lugar de tenerlos perfectamente organizados en diferentes carpetas de armarios.

Por complicado que parezca, estos documentos no tienen un esquema predefinido como las bases de datos relacionales, lo cual es perfecto para las aplicaciones de IoT actuales y futuras. Esto también ayuda a distribuir los datos entre varios servidores, lo que lo hace más amigable con la nube.

Volviendo a nuestro ejemplo, los datos sobre cada pozo ahora se almacenan en un documento y no en una tabla:

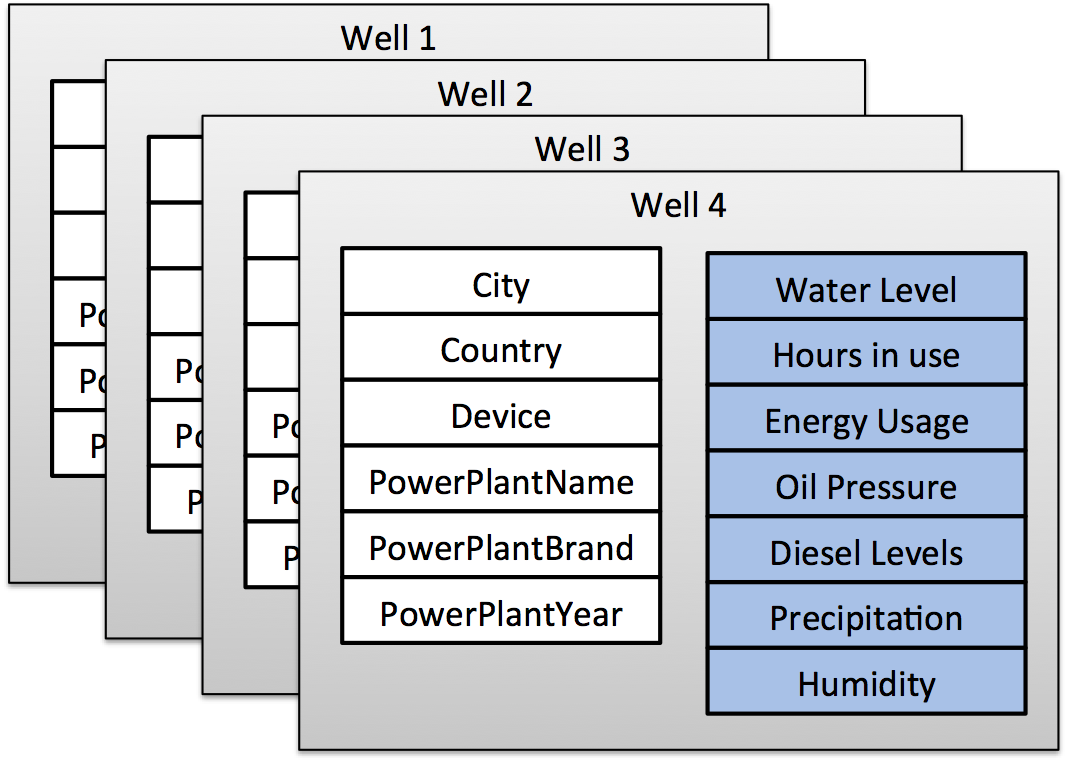


Ilustración 8 – Arquitectura para medir el agua de un pozo más generadores diésel  
formato documento

Como puede ver, fusionamos todas las propiedades del pozo en un solo documento, independientemente de las relaciones entre el pozo (nivel de agua, precipitación, humedad), su entorno (ciudad, país) y el generador de energía (presión de petróleo, uso de energía, etc.).

Esta simplificación permite cualquier cambio futuro de las propiedades del documento.

Finalmente, la elección de MongoDB para el proyecto se fundamenta en su capacidad para manejar datos de manera flexible y escalable, así como en su capacidad para adaptarse a cambios en el esquema de datos sin una reescritura extensa. Además, se basa en el éxito comprobado de la API de Ubidots [23], que utiliza MongoDB, y en la experiencia previa de la organización con esta tecnología, lo que respalda su idoneidad para el proyecto en cuestión.

### Amazon Web Services EC2

Amazon Web Services Elastic Compute Cloud (EC2) [24] es un servicio líder en la industria de la computación en la nube que ofrece capacidades de procesamiento escalables en la infraestructura de Amazon. Con EC2, los usuarios pueden alquilar máquinas virtuales (VM) en la nube para ejecutar aplicaciones de manera eficiente y escalable, aprovechando una amplia gama de instancias de VM con diferentes recursos, como CPU, memoria, almacenamiento y capacidad de red.

Una característica distintiva de EC2 es su capa gratuita, que brinda recursos limitados de forma gratuita durante un período de tiempo específico para nuevos usuarios. Esta opción permite a los usuarios experimentar y familiarizarse con la plataforma de AWS sin incurrir en costos iniciales significativos.

En el contexto de este proyecto, se optará por utilizar la capa gratuita de EC2 [25] en primera instancia para implementar y probar el proyecto. Esta decisión se basa en la necesidad de una solución rentable para alojar y ejecutar la aplicación inicialmente, permitiendo su desarrollo y prueba sin incurrir en gastos considerables de infraestructura. Además, la flexibilidad de los precios de Amazon garantiza que solo se pagará por los recursos utilizados, evitando gastos innecesarios en sobreprecios.

# Arquitectura Final de LIITEC API

Basándose en la arquitectura, tecnologías y herramientas previamente descritas y evaluadas, se presenta la estructura propuesta para la LIITEC API. Esta arquitectura está diseñada específicamente para abordar la necesidad y requisito del Laboratorio LIITEC en cuanto a recolección y gestión de datos para el monitoreo de variables ambientales.

La estructura central sigue los principios fundamentales de extracción, transformación y carga (ETL), adaptados a los objetivos particulares del laboratorio.

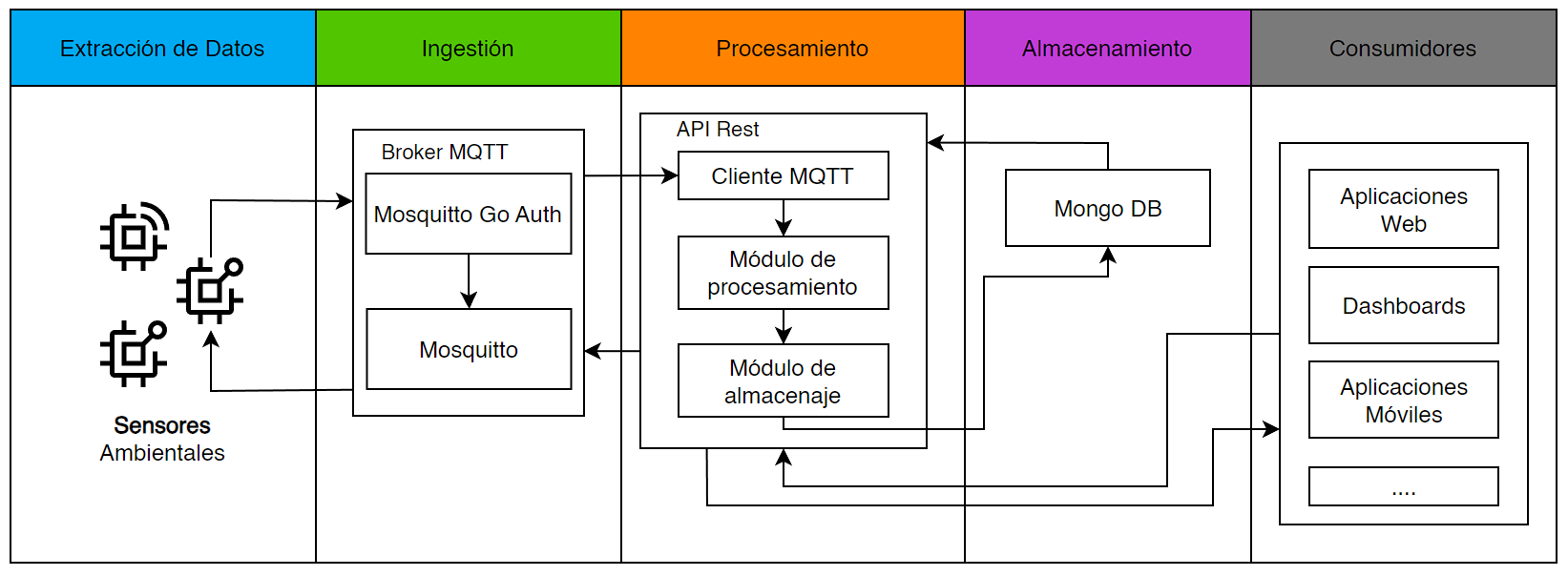


Ilustración 9 - Arquitectura de datos LIITEC API

La imagen muestra un diagrama de flujo de datos de la arquitectura de LITEC API. Este se divide en cinco etapas principales: Extracción de Datos, Ingestión, Procesamiento, Almacenamiento y Consumidores. Así es como los datos fluyen a través del sistema:

1. **Extracción de Datos:** Los datos se extraen de los "Sensores Ambientales". Estos sensores pueden ser de varios tipos, como sensores de temperatura, humedad, presión, etc., dependiendo de la aplicación. Los datos de estos sensores se recogen y se preparan para la siguiente etapa.
2. **Ingestión:** Los datos de los sensores se envían a un "Broker MQTT", que es un servidor que recibe mensajes de los sensores y luego los distribuye a los clientes que están suscritos a cierto *tópico*. El "Broker MQTT" en este caso incluye "Mosquitto Go Auth" para autenticación y "Mosquitto" como el broker MQTT. Esto asegura que sólo los clientes autorizados puedan recibir los datos.
3. **Procesamiento:** Los datos se pasan a una "API Rest" que interactúa con un "Cliente MQTT". La API Rest es una interfaz que permite a las aplicaciones interactuar con el sistema y solicitar los datos. El "Módulo de procesamiento" procesa los datos, lo que podría incluir limpieza de datos, transformación, análisis, etc. Después del procesamiento, los datos se envían al "Módulo de almacenaje".
4. **Almacenamiento:** Los datos procesados se almacenan en "Mongo DB", una base de datos NoSQL.
5. **Consumidores:** Finalmente, los datos almacenados están disponibles para los consumidores, que pueden ser "Aplicaciones Web", "Dashboards" y "Aplicaciones Móviles", entre otros. Estos consumidores pueden solicitar los datos de la base de datos y utilizarlos para una variedad de propósitos, como análisis, visualización, toma de decisiones, etc.

[Arquitectura de productos de plataforma de IoT en Google Cloud  |  Cloud Architecture Center](https://cloud.google.com/architecture/connected-devices/iot-platform-product-architecture?hl=es-419)

[¿Qué es la arquitectura IoT y dónde se emplea? | UNIR](https://www.unir.net/ingenieria/revista/arquitectura-iot/#:~:text=La%20arquitectura%20IoT%20es%20un,donde%20ser%C3%A1n%20procesados%20y%20almacenados.)

# Bibliografía

[1] [NUEVO PLAN DE EMERGENCIA HÍDRICA PONE SUS ÉNFASIS EN EL ABASTECIMIENTO HUMANO, DESARROLLO PRODUCTIVO Y APOYO SOCIAL - GORE Coquimbo](https://www.gorecoquimbo.cl/nuevo-plan-de-emergencia-hidrica-pone-sus-enfasis-en-el-abastecimiento/gorecoquimbo/2019-12-13/120714.html#:~:text=La%20Regi%C3%B3n%20de%20Coquimbo%20enfrenta,y%20a%20la%20econom%C3%ADa%20familiar%20campesina.)

[2] [Cambio climático: Chile sumará 14 años consecutivos de sequía y proyectan aumento en olas de calor – MMA](https://mma.gob.cl/cambio-climatico-chile-sumara-14-anos-consecutivos-de-sequia-y-proyectan-aumento-en-olas-de-calor/)

[3] [CEAZA pronostica altas temperaturas para valles interiores de la Región de Coquimbo - Ceaza](http://www.ceaza.cl/2024/01/29/ceaza-pronostica-altas-temperaturas-valles-interiores-la-region-coquimbo/)

[4] bcs365, "Third-party apps: what are the risks?". URL: <https://bcs365.com/insights/third-party-apps-what-are-the-risks>. Fecha de acceso: 11/12/2023.

[5] Carrie J., Samuel R., Bryn G. “The technological advancements that enabled the age of big data in the environmental sciences: A history and future directions”. URL: [The technological advancements that enabled the age of big data in the&nbsp;environmental sciences: A history and future directions (sciencedirectassets.com)](https://pdf.sciencedirectassets.com/315525/1-s2.0-S2468584420X00055/1-s2.0-S2468584420300489/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjELz%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIEN5M5%2FAXNz%2BMN6oExKxkwMjJ3r5mt7bRFbRB6Lf%2BWTzAiEAvPv%2FCIOZyL%2F6%2BjxMc2bPG6%2BjB79YZD1V%2BsnL5j%2F3GT4qswUINRAFGgwwNTkwMDM1NDY4NjUiDF5EhuJZz%2F82XmU1iyqQBdiRxMWWUSQ2gBFgUFh5DgiUU4TLhDV2tnGiMKyW09dMRqq17VunofgBpcgc8RXAloqEX6uUysCZlwxtXuDUrJLGOLuYwJJIir%2FOVMm0NxgxX7y8ocd0NIfDTsm2TLkEAn%2FwIe8bge7so91v6IqpaPTY%2BhO1hZdLWMs%2FrU%2BraZvyqQjPnDdPhykPSKi6ZKJ6KzO47I8MkTZZktU3iQiLOU8d114G26oOQ9oGbDH0iBNofsjk9mK4fXwkJohajKBCjIQ%2ByMBn1CBlwm%2FS9kyvA3%2BfV5MY56PZw%2Fl9nghos5suPD3F8XhKntshgO104iqsc3NuGpuhpy5bPH9jyxpAzFXCVRWW%2F86tsv0gm6PVaXrE%2B%2FoMF2MEvxi5qTSyPnK6ywO%2F%2B3khrYaSE4oSj5Wu54VNVEjJY4Gm2wDr9ck5zH3iUtttDI%2FgPXIisEErdANbcSgId0H4ULZbqHslMkTf1K49geFkAzcaV6uGpE9JtRDQ4k7%2Biz6gR44SimUCN7ib0t0QOzKcgpsCzOe0FFwSTBD%2Be8TsGQkquqLDAa5kJ4OD8XoXjaNc44c8KEcxGpwqnKsFfSGsgauaiNF%2BSiKW6MYMcr8oErmxHPFz2ukqM3OtPh3LD5Nt%2FqbwiCujRj5z19JRWl6IAqfwzuWSCGKlqcUAdfUzuwG8aKcNkNXQtejo%2FLROJMQHzQwiLivp3MD4OtJs5WnUGYQsQEHKKAwO7MP8orH7CQaK7aCHsRfJ6aYuLezuucozOjZCO1fXeR14VWJTUbLOC3HNWiXsufF1y2NOK6WPCdYsrc9tGRTtBH9WUlE7OMMx9Q%2BIBkaUix7GY33ZShJh%2BkrV7RNJTALTXyUraPcrFx7P8uGTKXjrDD8MMJHw4qsGOrEBZFkIXCzE%2FcAZfPXGMg0wouoMqvdzA6LxxYrOOlQZrqAuNbYubd28UmLylYiyFL2ebTifCpFm2%2B1V%2BDPEspmjENsEBof3Ls8novVRVpYzBi%2BuYReDjMoxw8oKbkWTmNkCtReKqEV8I30nwpVdqqpnusXKGeArIKNLQQiKMKEVGFlVDzIXyH0ehJtXpxOgOC31MnTEzMH2NDsXax%2Fmuu3bHHBw65PLgVzRp0KuuOuj2InT&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20231212T194949Z&X-Amz-SignedHeaders=host&X-Amz-Expires=300&X-Amz-Credential=ASIAQ3PHCVTY3CRLZI62%2F20231212%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=347094df799c15506153b669323b511e486be5bbf77daa28f9fc24e0efaa2844&hash=b91fde2b03bfe40f82e8b710821b361e0441e440a5ec721ee1ec94e431045772&host=68042c943591013ac2b2430a89b270f6af2c76d8dfd086a07176afe7c76c2c61&pii=S2468584420300489&tid=spdf-1eff6950-3308-497b-921b-66c1b5fb57de&sid=2fc2b725400f66418d786681ee22f310116cgxrqa&type=client&tsoh=d3d3LnNjaWVuY)

[6] [What is an Application Programming Interface (API)? | IBM](https://www.ibm.com/topics/api)

[7] [How APIs Can Transform Your Company (forbes.com)](https://www.forbes.com/sites/falonfatemi/2019/03/21/how-apis-can-transform-your-company/#bbbde7d668c2)

[8] [API Gateway - IBM API Connect](https://www.ibm.com/products/api-connect/api-gateway)

[9] [Build a Real-time ETL Pipeline for an IoT System | Krasamo](https://www.krasamo.com/etl-pipeline/)

[10] [Procedimientos recomendados para el diseño de API web - Azure Architecture Center | Microsoft Learn](https://learn.microsoft.com/es-es/azure/architecture/best-practices/api-design)

[11] [Diseño orientado a recursos  |  API de Cloud  |  Google Cloud](https://cloud.google.com/apis/design/resources?hl=es-419)

[12] [Arquitectura de una API REST · Desarrollo de aplicaciones web (gitbooks.io)](https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html)

[13] [What is Docker? | IBM](https://www.ibm.com/topics/docker), [What is Docker? (redhat.com)](https://www.redhat.com/en/topics/containers/what-is-docker)

[14] [Mosquitto MQTT Broker: Pros/Contras, Tutorial y una Alternativa Moderna | EMQ (emqx.com)](https://www.emqx.com/en/blog/mosquitto-mqtt-broker-pros-cons-tutorial-and-modern-alternatives), [What is Eclipse Mosquitto (MQTT)? (educative.io)](https://www.educative.io/answers/what-is-eclipse-mosquitto-mqtt), [Mosquitto : MQTT. Overview | by Bhagvan Kommadi | Medium](https://medium.com/@bhagvankommadi/mosquitto-mqtt-2a352bd8f179)

[15] [¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS (amazon.com)](https://aws.amazon.com/es/what-is/mqtt/)

[16] [Calidades de servicio proporcionadas por un cliente de MQTT - Documentación de IBM](https://www.ibm.com/docs/es/ibm-mq/9.1?topic=concepts-qualities-service-provided-by-mqtt-client)

[17] [mosquitto.conf man page | Eclipse Mosquitto](https://mosquitto.org/man/mosquitto-conf-5.html), [Complemento de seguridad dinámica | Eclipse Mosquitto](https://mosquitto.org/documentation/dynamic-security/)

[18] [iegomez/mosquitto-go-auth: Plugin de autenticación para mosquitto. (github.com)](https://github.com/iegomez/mosquitto-go-auth)

[19] [¿Qué es Express.js? Todo lo que Debes Saber (kinsta.com)](https://kinsta.com/es/base-de-conocimiento/que-es-express/) , [Introducción a Express/Node - Aprenda desarrollo web | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction#how_popular_are_node_and_express)

[20] [EasyShopC2C-Application/RestAPI-EasyShop at main · Arlezz/EasyShopC2C-Application (github.com)](https://github.com/Arlezz/EasyShopC2C-Application/tree/main/RestAPI-EasyShop)

[21] [Companies using Express (expressjs.com)](https://expressjs.com/en/resources/companies-using-express.html)

[22] [Diseño para el Internet de las Cosas (ubidots.com)](https://ubidots.com/blog/designing-for-the-internet-of-things/#:~:text=At%20Ubidots%2C%20it's%20been%20our,Variables%20that%20contain%20Values)

[23] [Welcome (ubidots.com)](https://docs.ubidots.com/reference/welcome)

[24] [AWS | Elastic compute cloud (EC2) de capacidad modificable en la nube (amazon.com)](https://aws.amazon.com/es/ec2/)

[25] [Capa gratuita de AWS | Cloud computing gratis |AWS (amazon.com)](https://aws.amazon.com/es/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all)

1. ThingSpeak, "ThingSpeak: Plataforma de Análisis IoT", MathWorks. URL: <https://la.mathworks.com/products/thingspeak.html> [↑](#footnote-ref-1)
2. Big Data: Grandes volúmenes de datos, difíciles o imposibles de procesar a través de métodos tradicionales, que sirven para obtener ideas y tomar mejores decisiones de negocios. [↑](#footnote-ref-2)
3. ThingSpeak, “License Options”, URL: <https://thingspeak.com/prices> [↑](#footnote-ref-3)
4. Ubidots, “Plans Capacity”, URL: [https://es.ubidots.com/pricing - plans-capacity](https://es.ubidots.com/pricing#plans-capacity) [↑](#footnote-ref-4)
5. Blynk, “Pricing Plans”, URL: <https://blynk.io/pricing> [↑](#footnote-ref-5)
6. Losant, “Resource Limits”, URL: <https://docs.losant.com/organizations/resource-limits/> [↑](#footnote-ref-6)
7. Stateless: no tienen una dependencia de almacenamiento del lado del servidor para los estados del cliente. [↑](#footnote-ref-7)
8. Cloud Native: [Cloud Native | IBM](https://www.ibm.com/topics/cloud-native) [↑](#footnote-ref-8)
9. Multicloud: [What is Multicloud? | IBM](https://www.ibm.com/topics/multicloud) [↑](#footnote-ref-9)