Toma de decisiones a través de un dashboard integrado en la nube para el sistema de monitoreo y prevención de incendios forestales

Freire Peñafiel, Roberto Patricio y Ochoa Maldonado, Sebastián Alejandro

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Trabajo de integración curricular, previo a la obtención del título de Ingeniero en Tecnologías de Información

Marcillo Parra, Diego Miguel, PhD

25 de agosto del 2023

Copyleaks

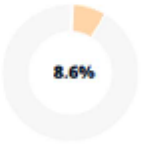
Plagiarism and AI Content Detection Report

TrabajodeIntegracionCurricularRevis...

Scan details

Scan time:	Total Pages:	Total Words:
August 22th, 2023 at 12:38 UTC	48	11995

Plagiarism Detection



Types of plagiarism		Words
Identical	0.4%	47
Minor Changes	0.3%	32
Paraphrased	7.3%	874
Omitted Words	7.4%	891

Firma:

.....

Ing. Marcillo Parra Diego Miguel PhD

Director.

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Certificación

Certifico que el trabajo de integración curricular "**Toma de decisiones a través de un dashboard integrado en la nube para el sistema de monitoreo y prevención de incendios forestales**" fue realizado por los señores **Freire Peñafiel, Roberto Patricio y Ochoa Maldonado, Sebastián Alejandro**, el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 23 de octubre de 2023

.....

Ing. Marcillo Parra Diego Miguel
PhD

C. C. 1710802925

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Responsabilidad de Autoría

Nosotros, **Freire Peñafiel, Roberto Patricio y Ochoa Maldonado, Sebastián Alejandro**, con cédulas de ciudadanía n° 1721229399 y 1726134503, declaramos que el contenido, ideas y criterios del trabajo de integración curricular: **Toma de decisiones a través de un dashboard integrado en la nube para el sistema de monitoreo y prevención de incendios forestales** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 19 de octubre de 2023

Freire Peñafiel, Roberto Patricio
C.C.: 1721229399

Ochoa Maldonado, Sebastián Alejandro
C.C.: 1726134503

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Autorización de Publicación

Nosotros, **Freire Peñafiel, Roberto Patricio y Ochoa Maldonado, Sebastián Alejandro**, con cédulas de ciudadanía n° 1721229399 y 1726134503, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Toma de decisiones a través de un dashboard integrado en la nube para el sistema de monitoreo y prevención de incendios forestales** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Sangolquí, 19 de octubre de 2023

Freire Peñafiel, Roberto Patricio
C.C.: 1721229399

Ochoa Maldonado, Sebastián Alejandro
C.C.: 1726134503

Dedicatoria

El presente trabajo lo dedico principalmente a mis padres, por haber sido ese ejemplo de dedicación, excelencia y resiliencia a lo largo de mi vida, como ellos siempre me han dicho, lo mejor que me pueden dejar es el estudio y con este gran paso estoy convencido de que así lo es. Por esto reciban este trabajo de titulación como muestra de agradecimiento a todo el esfuerzo y sacrificio que han hecho por mi hermana y por mí.

A mis abuelitos Luis y Consuelo, que nunca faltaron a su titulo de abuelos y supieron como complementar mi crianza en todas las etapas de mi vida, inclusive hasta hoy sigo aprendiendo de su humanidad, sencillez y amor incondicional. Les dedico este trabajo porque son parte importante para que esta meta se haya alcanzado.

A mi hermana Samantha; por apoyarme y creer en mí. Este trabajo te lo dedico esperando que siembre en ti la espinita de seguir adelante con tu carrera y ser la excelente profesional que sin lugar a duda estoy seguro de que vas a llegar a serlo.

Sebastián Ochoa

A mis padres Roberto y Consuelo, y mi hermana Sofía, quienes permitieron que se cumpla este sueño, gracias al apoyo incondicional que me brindaron durante todo este tiempo, a sus consejos y paciencia. Ellos siempre serán el pilar fundamental en mi vida, les dedico este trabajo. A mi pareja Carolina que ha sido una fuente de motivación, a mis amigos y compañeros que han sido el mejor regalo que me ha brindado esta experiencia universitaria; sobre todo a Wilmer, quien más que amigo ha sido un hermano.

Roberto Freire

Agradecimiento

Por el presente trabajo, agradezco a Dios y la Virgen, por estar cuando nadie más estuvo, por escuchar mis oraciones, pero sobre todo por saber iluminar mi camino y guiarme en esta gran travesía. A mis padres y a mi hermana, por ser el equipo que somos. Gracias a su apoyo, comprensión y cariño he logrado culminar una etapa más. A mi tío Gabriel por haber sido un apoyo en este caminar, por cada plática en la que me hacía entender la vida desde otros puntos de vista, por haber sido como un segundo padre para mí, gracias ñaño este logro lo comparto contigo. A mi compañero que fue parte de este trabajo, gracias por la paciencia y la colaboración prestada no solo en este proyecto, sino a lo largo de la carrera. Colega Roberto éxitos en tu vida profesional, te lo mereces.

Sebastián Ochoa

Agradezco a la Universidad de las Fuerzas Armadas ESPE por todos los recursos invertidos en la formación académica de mi persona, a sus docentes y personal que aportaron en este proceso académico. A mi compañero de tesis, Sebastián, el cual ha sido un eje fundamental en el desarrollo de este proyecto.

Finalmente, a nuestro tutor de tesis, Diego Marcillo Ph. D., sin su formación y consejo este trabajo no sería posible.

Roberto Freire

Índice de contenidos

Índice de tablas

Índice de figuras

Resumen.

Los incendios forestales son desastres naturales que generan graves consecuencias tanto para la biodiversidad del lugar como para el medio ambiente en general. Debido a esto una de las tareas más importantes es el poder tomar decisiones de forma acertada y oportuna. Mientras tanto el avance de la tecnología en el ámbito de la información y la comunicación ha permitido obtener grandes avances al momento de reunir, examinar y procesar grandes cantidades de información, debido al desarrollo de tecnologías como la computación en la nube, el internet de las cosas IoT, y el manejo de grandes cantidades de información conocido como Big Data. EL presente trabajo identifica la oportunidad de utilizar estas tecnologías en el desarrollo de una aplicación cuyo fin sea el recopilar y desplegar información ambiental en tiempo real a través de un dashboard web, para el análisis y toma de decisiones con el objetivo de mejorar la gestión y prevención de incendios forestales. El presente trabajo se realizó un análisis de las tecnologías necesarias requeridas para el desarrollo del prototipo. Se desarrolló un prototipo el cual incorporó un nodo de sensores IoT y la transmisión de video en tiempo real con la emulación de una cámara de drone, implementado en una infraestructura cloud. Los resultados demostraron la funcionalidad del prototipo el cual permitió obtener la información en tiempo real de los sensores, el almacenamiento de esta información en la nube y el despliegue de la misma en el dashboard web, así como la transmisión del video en tiempo real.

Palabras clave: incendios forestales, computación en la nube, internet de las cosas, dashboard web, big data.

Abstract.

Forest fires are natural disasters that generate serious consequences both for the biodiversity of the area and for the environment in general. Because of this, one of the most important tasks is to make accurate and timely decisions. Meanwhile, advances in information and communication technology have allowed for significant progress in gathering, examining, and processing large amounts of information, thanks to the development of technologies such as cloud computing, the Internet of Things (IoT), and the handling of large amounts of information known as Big Data. This work identifies the opportunity to use these technologies in the development of an application whose purpose is to collect and display real-time environmental information through a web dashboard for analysis and decision-making with the aim of improving the management and prevention of forest fires. In this work, an analysis of the necessary technologies required for the development of the prototype was carried out. A prototype was developed that incorporated an IoT sensor node and real-time video transmission emulating a drone camera, implemented in a cloud infrastructure. The results demonstrated the functionality of the prototype, which allowed for real-time information from the sensors, the storage of this information in the cloud, its display on the web dashboard, and real-time video transmission.

Keywords: microservices, Results Management System, connection with society, investigation, teaching.

Capítulo I: Generalidades

Tema

Toma de decisiones a través de un dashboard integrado en la nube para el sistema de monitoreo y prevención de incendios forestales

Antecedentes

Cuando se habla acerca de la prevención de los incendios forestales, una de las tareas más importantes es el poder tomar decisiones de forma acertada y oportuna, debido a que este tipo de desastre natural conlleva severas consecuencias tanto para la biodiversidad del lugar como para el medio ambiente en general, ya que no solo malogra los recursos naturales, sino que también afecta la calidad de vida de las comunidades aledañas (Guerreo & Carolina, 2019). Debido a esto, es fundamental disponer de herramientas tecnológicas que favorezcan el poder tomar las mejores decisiones posibles con el fin de detener o prevenir dichas consecuencias (Gerrero & Carolina, 2019).

Con el avance de la tecnología en el ámbito de la información y comunicación que se ha llevado a cabo en los últimos años, se ha logrado una gran evolución al momento de reunir, examinar y exponer la información importante para una oportuna toma de decisiones. Uno de los elementos que más ha sobresalido de dicha evolución es la creación de los dashboards integrados a la nube, cuyo potencial de efectividad ha sido evidenciado en diferentes ámbitos como la gestión y monitoreo empresarial, o el análisis de datos en tiempo real (Bogotá et al., 2023).

Los dashboards son interfaces de tipo visual que reúnen cantidades inmensas de información dentro de paneles interactivos, cuyos resultados no presentan complicaciones para ser interpretados. Estas interfaces posibilitan a sus usuarios el examinar y crear análisis de forma instintiva a partir de datos complejos, facilitando una amplia visión clara y precisa para identificar fácilmente los patrones y las tendencias acerca de la situación actual (Myatt &

Johnson, 2009). A su vez, debido a que dichos dashboards están integrados a la nube, se suprimen las barreras geográficas ya que se puede acceder a ellas desde cualquier dispositivo y desde cualquier lugar del mundo, logrando que cualquier actor involucrado tenga acceso instantáneo a la información allí guardada.

Los incendios forestales son acontecimientos con bajo potencial de predictibilidad, ya que para ello se requiere una vigilia constante de diferentes variables como la dirección del viento, temperatura, humedad, extensión y dispersión del fuego. A su vez, resulta indispensable disponer de información geoespacial actual y tener a la mano los planos de las áreas afectadas, así como las posibles rutas de escape (Guerrero & Catolina, 2019). En este caso, el empleo de dashboards integrados en la nube facilitaría la toma de decisiones para gestionar y prevenir los incendios forestales ya que, aparte de brindar utilidades operativas, mejoraría las habilidades estratégicas debido a que la compilación y examinación sistemática de información a largo plazo posibilita establecer patrones estacionales, valorar la eficiencia de los métodos de prevención utilizados y perfeccionar los mecanismos de respuesta (Bogotá et al., 203).

Objetivos

Objetivo general

Crear el prototipo de un sistema de monitoreo gestión y prevención de incendios forestales en una aplicación web, a través de la integración de datos conseguidos mediante sensores y drones, empleando herramientas de desarrollo de software actuales para garantizar la eficacia y seguridad de la integración, con el objetivo de mejorar la gestión y prevención de este tipo de desastres.

Objetivos específicos

- Buscar información relevante acerca de metodologías y herramientas de desarrollo de software para el levantamiento del dashboard.

- Fabricar un prototipo de sistema que posibilite incorporar en tiempo real la lectura de nodos de sensores y visión de una emulación de cámara de un dron.
- Realizar el despliegue e integración del prototipo en la nube con el objetivo de lograr visualizar la información obtenida de manera global, utilizando diversas plataformas de despliegue para todas las capas del sistema.
- Valorar la correcta funcionalidad del sistema, la obtención y el registro histórico de la información.

Alcance

El presente trabajo se focaliza en el desarrollo del prototipo de una aplicación apoyada con tecnologías IoT (Internet de las cosas) y API REST (Interfaz de programación de aplicaciones con transferencia de estado representacional) para la oportuna toma de decisiones mediante de una aplicación web de tipo dashboard integrado en la nube, con el fin de optimizar el sistema de monitoreo para la prevención o detección temprana de incendios forestales. La aplicación web creada dará paso a la integración de los datos obtenidos a través de drones y sensores, los cuales a su vez serán utilizados para el desarrollo de modelos predictivos para la detección oportuna de incendios forestales.

El objetivo principal de la interfaz es brindar, a través de un dashboard, un reporte en tiempo real, con la finalidad de que el usuario pueda examinar la información compilada y ejecute protocolos cuando detecte incendios forestales. Para ello, se aplicó una arquitectura IoT, en la cual se utilizaron sensores y emulación de drones con cámara para recoger información de importancia, así como datos ambientales y streaming en tiempo real captado por la cámara del dron. Estos artefactos fueron conectados a través de una red inalámbrica y envían los datos obtenidos a un servidor central en tiempo real. La interfaz web fue desarrollada a través del uso de herramientas de desarrollo de software modernas y se integró con la arquitectura IoT a través de tecnologías API REST. Esto posibilita una emisión eficaz y

confiable de los datos entre los dispositivos IoT y la plataforma de la nube donde está instalado el prototipo.

La utilización de API REST consolidará una comunicación segura y fluida, asegurando totalmente la disponibilidad de la información actualizada para su examinación y posterior toma de decisiones oportunas.

Resulta importante aclarar que este proyecto se focaliza en el diseño y desarrollo de un prototipo del dashboard y la integración de los datos obtenidos por los drones y sensores, pero no incluye la creación de los dispositivos IoT, lo cual fue fundamentado en el proyecto propuesto porOrellana (et al., 2021). El enfoque se focaliza en la interfaz, la obtención de datos y la integración de dicha información; así como la proporción de una moderna y eficaz solución aprovechando las características de la tecnología IoT y la integración de datos en un dashboard en la nube para la prevención de incendios forestales.

Capítulo II: Marco Teórico

Introducción a la integración de datos en un dashboard web en la nube

Internet de las cosas IoT

Es un conjunto de tecnologías y protocolos que representa una importante transformación en la interconexión de objetos cotidianos mediante la infraestructura digital global. Concretamente, es una red de dispositivos físicos que poseen un software, sensores y tecnología conectiva que les posibilita compilar, cambiar y trabajar sobre datos.

La confluencia de la conectividad de banda ancha, la tecnología de sensores y la examinación de datos permitió la fabricación de ecosistemas conectados unos con otros, cuyos objetos pueden comunicarse entre sí y, a su vez, con sistemas informáticos, logrando así una gran serie de aplicaciones funcionales y prácticas en diferentes ámbitos como la salud, industria manufacturera, logística, el hogar inteligente, etc. El IoT tiene la capacidad de cambiar

a fondo la forma en la que las personas se interrelacionan con el entorno, mejorando los procesos y la toma de decisiones, logrando el alcance de resultados innovadores para tratar los retos de la actualidad (Talwana & Hua, 2017).

The Things Network TTN

Se trata de una red inalámbrica de comunicación que posee baja potencia, cuyo objetivo principal es la interconexión de dispositivos en el marco de la Internet de las Cosas. TTN tiene sus fundamentos en una tecnología llamada LoRaWAN (Long Range Wide Area Network) la cual es una comunicación de largo alcance y de bajo consumo energético; a su vez, se ha transformado en la principal solución para facultar la conectividad de dispositivos IoT dentro de áreas urbanas y rurales, posibilitando de esta forma el envío de datos en ambas direcciones y a largas distancias, manteniendo una vida útil extendida de batería.

The Things Network se ha posicionado como el principal constructor de redes descentralizadas que favorece el desarrollo de aplicaciones y servicios vanguardistas, esto gracias a su focalización en la arquitectura de código abierto junto con la colaboración comunitaria. Las características que ofrece esta red han sido aprovechadas en varios sectores que van desde la agricultura inteligente hasta la atención de salud remota (Suwaid et al., 2019).

En el siguiente marco teórico se examinarán los principios fundamentales de TTN, así como su función en IoT; a su vez se verificarán sus probables aplicaciones em varias áreas de la tecnología y la sociedad, y cuáles son sus ventajas y desventajas.

Cloud Computing

También llamado computación en la nube, es un sistema tecnológico que ha transformado la manera en que los usuarios acceden, guardan y gestionan los medios informáticos, ya que son capaces de proveer diversos servicios que van desde el almacenamiento a gran escala, a procesamiento de aplicaciones mediante internet. Este

sistema se adapta fácilmente a las demandas actuales ya que posibilita a los individuos acceder de forma sencilla a recursos escalables debido a que suprime la necesidad de poseer una infraestructura física de alto costo.

La nube posee diferentes formatos de servicio que brindan varios niveles de abstracción y control, entre los cuales están Infrastructure as a Service (IaaS), Platform as a Service (PaaS) y Software as a Service (SaaS). Este sistema no solo ofrece beneficios en torno a la eficiencia y agilidad, sino también una alta capacidad de seguridad y privacidad de los datos. Por ello, a continuación, se examinarán las bases y modelos del cloud computing, así como su gran influjo en la forma en que los usuarios ejecutan los recursos tecnológicos de la nueva era (Qian et al., 2009).

Tecnologías de desarrollo web

Este tipo de tecnologías posibilitan a los desarrolladores el poder crear aplicaciones web dinámicas y actuales ya que poseen lenguajes de programación (JavaScript), hojas de estilo (CSS) y lenguajes de marcado (HTML), lo cual ofrece al usuario una experiencia más amigable y fluida (Ibarra et al., 2021).

Frameworks para el Desarrollo de Software

Estos son grupos de herramientas, bibliotecas y componentes que, gracias a su previa definición, facilitan el desarrollo, incorporación y conservación de sistemas y aplicaciones de software. Estos frameworks posibilitan a los desarrolladores focalizarse en la lógica de negocio, debido a que brindan una estructura y un grupo de normas que reducen la necesidad de escribir código repetitivo, logrando la construcción de un software más eficiente (Pantoja & Pardo, 2016).

API

Llamada Interfaz de programación de aplicaciones, se refiere a un grupo de normas y matrices que posibilita la intercomunicación entre varias aplicaciones de software.

Las APIs permiten la incorporación y el intercambio de datos de forma eficiente y segura, por lo que se dice que han transformado la manera en que las aplicaciones y los servicios se conectan y colaboran entre sí (De, 2017).

API REST. Sus siglas significan Interfaz de Programación de Aplicaciones en Transferencia de Estado Representacional, y han surgido como un modelo preeminente en la arquitectura de aplicaciones actuales para el desarrollo y presentación de servicios web. Los fundamentos principales de API son los preceptos del protocolo HTTP y se valen de los formatos estándar tales como GET, POST, PUT y DELETE con el fin de posibilitar la intercomunicación y envío de datos entre diferentes sistemas. Por otro lado, la principal base de REST es la representación de recursos y su estado a través de URLs, posibilitando una comunicación lineal y congruente. A través del manejo de protocolos de intercambio de datos tales como XML o JSON, las API REST permiten el desarrollo de aplicaciones flexibles y con alto potencial de escalamiento, las cuales pueden ser integradas a varias plataformas o dispositivos. Cabe recalcar que se necesitan de consideraciones especificas acerca de la seguridad, la autenticación, el versionado y la definición de rutas para que la implementación de una API REST sea favorable. Es por esto por lo que, a continuación, se analizarán los principios básicos de las API REST, así como su función en el desarrollo de interfaces interoperables, y su importancia en la arquitectura de sistemas distribuidos para lograr una correcta comunicación entre aplicaciones (De, 2017).

Bases de datos

Estos sistemas posibilitan el acceso y el manejo competente de amplios grupos de datos ya que poseen protocolos de almacenamiento estructurado de información. Gracias a ello son capaces de proveer alta velocidad, seguridad y escalabilidad, con el fin de sacias las exigencias de los usuarios de la nueva era (Beynon, s/f).

Bases de datos NoSQL. Sus siglas significan Not Only SQL y han sido creadas como una opción avanzada a los sistemas de administración de bases de datos, sobre todo en circunstancias en donde los requerimientos de almacenamiento y restauración de datos son escalables. Este tipo de base de datos acogen una amplia gama de modelos de datos flexibles, tales como columna-gráfico, documento o clave-valor. Esta característica los diferencia de las bases de datos relacionables cuyas bases son esquemas fijos y estructuras tabulares. Los modelos NoSQL son altamente beneficiosos para los ambientes Big Data y aplicaciones de alta recurrencia, debido a que posibilitan a las aplicaciones el guardar y manipular datos semi estructurados o no estructurados. Cabe recalcar que, a pesar de sus excelentes características en torno a la escalabilidad horizontal, flexibilidad y rendimiento, una desventaja que presentan las bases de datos NoSQL es la poca precisión en ámbitos como la integridad, la consistencia y las consultas complejas. Sin embargo, se examinarán los fundamentos de este tipo de bases de datos, sus características positivas y negativas, y su competitividad para diferentes modos de uso; de esta forma se dejará en evidencia que dichas tecnologías están revolucionando la manera en la que se acceden y guardan los datos en el contexto actual (del Busto & Enríquez, 2012).

MongoDB Atlas. Se refiere a un servicio de base de datos en la nube que brinda una solución holística para almacenar y gestionar con gran escalabilidad de bases de datos MongoDB. Cuando se habla de la creación de sistemas y aplicaciones, esta herramienta ejerce una función muy importante al brindar una organización robusta y completamente disponible

para el alojamiento de bases de datos NoSQL. A su vez, este servicio suprime la necesidad de cuidado especial al tratarse de labores operativas de alta complejidad, por lo cual posibilita al desarrollador el enfocarse en la creación de sistemas y aplicaciones, mejorando así su experiencia. Otra de las características especiales que ofrece esta base de datos es la replicación automática de datos, de tal forma que asegura una alta disponibilidad; a su vez, permite una automática copia de seguridad con el objetivo de mantener los datos estales; y finalmente, la alta escalabilidad horizontal para gestionar el aumento de datos (Mongo DB Documentation, s/f).

Visualización de datos

Este es el proceso encargado de representar los datos e información compleja a través de visualizaciones interactivas, diagramas o gráficos, con el fin de favorecer su entendimiento para su posterior análisis. Uno de sus beneficios es que permite al usuario identificar tendencias, patrones y relaciones de una manera más sencilla que los análisis tradicionales (VISUALIZACIÓN DE DATOS – Campus2B Bilbao, s/f).

Arquitectura de Software

Se trata de una agrupación de acuerdos y principios que orientan el diseño y la configuración de una aplicación. Esta es la encargada de demarcar la organización de cada elemento de un sistema, así como su interacción y define los requisitos funcionales de cada uno. Se debe tomar en cuenta que el hecho de elegir una arquitectura correcta es lo más importante para la creación de aplicaciones web; pues una arquitectura bien configurada favorecería el correcto desarrollo y crecimiento de la aplicación, ya que interfiere en varios ámbitos como la escalabilidad, modularidad, reutilización de código y la inoperabilidad. Dentro de los focos más comunes en del desarrollo de aplicaciones web se encuentran la arquitectura de tres capas (MVC) y la arquitectura orientada a servicios (SOA).

Arquitectura MVC

Este tipo de arquitectura es el modelo de diseño más usado para la creación de aplicaciones web, y se fundamenta en la segmentación de las responsabilidades de la aplicación en tres capas: modelo, vista y controlador (Fowler, 2003).

El modelo se encarga de gestionar los datos obtenidos, y de la aplicación de las reglas de negocio correspondientes, así como de la condensación de la estructura de datos, mientras brinda diferentes procedimientos para acceder, renovar o corroborar los datos registrados. El objetivo principal del modelo es garantizar la integridad de los datos y mantener la consistencia de la aplicación (Fowler, 2003).

La vista se refiere a un elemento de la arquitectura que se encarga de monitorear la visualización de la información al cliente (Grady, 2007). En síntesis, esta es la encargada de exponer la información del modelo, mostrar la interfaz y su interacción con el usuario final, a la vez que recoge las acciones del mismo. Esta vista se puede representar en una interfaz gráfica, una pantalla de aplicación móvil o una página web.

Finalmente, el controlador es quien recibe los pedidos del usuario a partir de la vista para posteriormente procesarlos, mientras organiza la intercomunicación entre modelo y vista e interpreta las acciones del usuario y las convierte en operaciones que manipulan los objetos del modelo (Pressman, 2010).

La arquitectura MVC ofrece varios beneficios en el desarrollo de aplicaciones web. Al separar las responsabilidades en diferentes componentes, se logra una mayor modularidad y reutilización de código. Esto facilita la evolución y el mantenimiento del sistema, ya que cada capa puede modificarse de manera independiente sin afectar las demás. Además, la arquitectura MVC promueve la escalabilidad y la flexibilidad al permitir que diferentes componentes se desarrollen y desplieguen de forma independiente.

Modelo. Esta capa se encarga de la lógica de negocio y la gestión de los datos. Aquí se definen las estructuras de datos, las reglas de negocio y las operaciones de persistencia. El modelo encapsula la información y proporciona métodos para acceder, manipular y validar los datos. En resumen, es la representación de la información con la cual el sistema opera (Fowler, 2003).

Vista. La capa de vista se ocupa de la presentación de la interfaz de usuario (Grady, 2007). Aquí se definen las plantillas y los elementos visuales que permiten a los usuarios interactuar con la aplicación. La vista se encarga de mostrar los datos del modelo y recoger las acciones del usuario, y puede utilizar lenguajes de marcado como HTML y tecnologías de plantillas para generar vistas dinámicas.

Controlador. Según lo señalado por Fowler en 2003 desempeña el papel de mediador entre el modelo y la vista. Recibe las solicitudes del usuario desde la vista, procesa la lógica de negocio correspondiente y coordina la interacción entre el modelo y la vista. El controlador maneja las acciones del usuario y actualiza el estado del modelo o la vista en consecuencia.

Esta arquitectura ofrece beneficios como la separación de responsabilidades, la reutilización de componentes y la facilidad de mantenimiento. Permite que diferentes partes de la aplicación evolucionen de forma independiente y promueve una mayor modularidad y escalabilidad.

Arquitectura Orientada a Servicios SOA

La arquitectura orientada a servicios (SOA) se basa en la creación de servicios independientes y autónomos que se comunican entre sí mediante interfaces bien definidas. En esta arquitectura, los servicios encapsulan la funcionalidad de negocio y se pueden acceder a través de protocolos estándar como REST (Representational State Transfer) o SOAP (Simple Object Access Protocol).

Los servicios en SOA se centran en ofrecer funcionalidades específicas y bien delimitadas, y se comunican mediante contratos claros. Esto permite la interoperabilidad de aplicaciones a través de la definición y el uso de servicios web basados en estándares abiertos (Chinnici et al., 2007) debido a que los servicios pueden ser desarrollados, actualizados y reemplazados de manera independiente.

Ambas arquitecturas tienen sus propias fortalezas y se pueden utilizar de manera complementaria.

Tabla 1

Fortalezas de las arquitecturas MVC y SOA

Arquitectura MVC	Arquitectura SOA
Separación clara de responsabilidades entre el modelo, la vista y el controlador, lo que facilita el mantenimiento y la escalabilidad del código.	Promueve la reutilización de servicios a través de interfaces estandarizadas, lo que permite construir aplicaciones flexibles y adaptativas.
Proporciona una estructura organizada y modular para el desarrollo de aplicaciones, lo que facilita la colaboración entre los equipos de desarrollo.	Permite la integración de sistemas heterogéneos y la interoperabilidad entre aplicaciones de diferentes plataformas y tecnologías.
Facilita la implementación de pruebas unitarias y de integración, lo que mejora la calidad del software y agiliza el proceso de desarrollo.	Permite la evolución independiente de los servicios, lo que facilita la actualización y el mantenimiento sin afectar a otras partes del sistema.
Es especialmente adecuada para aplicaciones web y de escritorio, donde se requiere una interfaz de usuario interactiva y una gestión eficiente de la lógica de negocio.	Ofrece una arquitectura flexible y adaptable a medida que los requisitos empresariales cambian y evolucionan.
	Permite la escalabilidad horizontal, agregando o eliminando servicios según las necesidades de rendimiento y demanda del sistema.

Nota. Esta tabla muestra las fortalezas de cada tipo de arquitectura de software utilizada para el desarrollo web, importante tener en cuenta que estas fortalezas son generales y pueden variar según el contexto y los requisitos específicos de cada proyecto

Arquitectura de microservicios

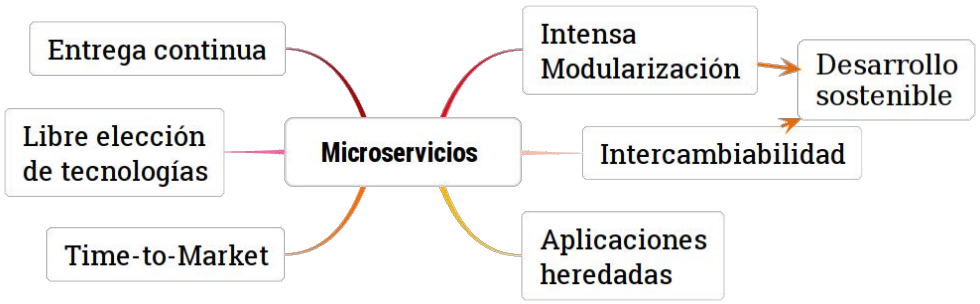
Es un enforque para el desarrollo de software en el que una aplicación se compone de un conjunto de servicios modulares (componentes/módulos) acoplados débilmente, donde cada componente cumple un objetivo comercial único. Esta arquitectura sigue el principio de una base de datos por cada servicio, lo que otorga bajo acoplamiento y permite al microservicio utilizar la base de datos que mejor se adapte a sus necesidades. La subdivisión de un gran sistema en microservicios facilita el mantenimiento y la evaluación del software, ya que cada microservicio puede ser manejado como un sistema independiente. Esto permite que un equipo se encargue del mantenimiento y la evolución de cada microservicio, lo que puede mejorar la calidad del software y reducir el tiempo necesario para realizar cambios(Tatiana Gómez Suárez et al., 2018).

La arquitectura de microservicios en la actualidad. La arquitectura de microservicios está ganando popularidad ya que se ve como una alternativa para evolucionar sistemas monolíticos, dado que dichos sistemas son cada vez más complejos, costosos para evolucionar, no cumplen las exigencias de escalabilidad y disponibilidad que exige el negocio(Tatiana Gómez Suárez et al., 2018).

A la vez este enfoque es el adecuado para el desarrollo de software que se despliega en la nube y puede tomar como ventajas la elasticidad y el aprovisionamiento de funcionalidades bajo demanda y el diseño de plataformas de IoT.

Figura 1

Beneficios de los microservicios



Nota: La figura representa cada uno de los beneficios que otorga el uso el estilo arquitectónico de microservicios para el desarrollo de aplicaciones web. Tomado de Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web (p5), por Daniel López, Edgar Maya. Séptima Conferencia de Directores de Tecnología de Información, TICAL 2017

Características

- Código base independiente: cada microservicio tiene su propio repositorio, lo que permite fácil mantenimiento y agilidad en el desarrollo.
- Heterogeneidad de tecnologías: cada microservicio se implementa utilizando su propia pila tecnológica, esto permite al equipo de desarrollo aprender a manejar diferentes tecnologías y evita la referencia a tecnologías innecesarias o grandes librerías.
- Escalabilidad independiente: cada microservicio puede ser escalado de forma independiente, lo que permite la fácil identificación de cuellos de botella y centra los esfuerzos del equipo en maximizar el rendimiento del mismo.
- Especialización funcional: el enfoque favorece la especialización funcional de cada equipo de desarrollo.
- Manejo de datos descentralizado: cada microservicio puede seleccionar la solución de persistencia que mejor se ajuste a su funcionalidad. La persistencia es necesaria para pocos microservicios (manejo de transacciones).

Ventajas y desventajas

Tabla 2

Ventajas y desventajas

Ventajas	Desventajas
Habilita el desarrollo e implementación continua de aplicaciones grandes y complejas.	Complejidad adicional al crear sistemas distribuidos.
Cada microservicio es relativamente pequeño.	Complejidad de despliegue.
Mejora en el aislamiento de fallas.	
Mejora en la capacidad de evolución.	

Nota. Esta tabla muestra las ventajas y desventajas que implican el uso del enfoque de la arquitectura de microservicios para el desarrollo de software.

Arquitectura Monolítica (MVC) vs arquitectura de microservicios

Es importante destacar que tanto MVC como la arquitectura de microservicios son enfoques válidos para el desarrollo de software, y la elección entre ellos dependerá de la naturaleza del proyecto, sus requerimientos, el equipo de desarrollo y otros factores relevantes.

Tabla 3

Comparativa entre la arquitectura monolítica MVC y la arquitectura de microservicios

Categoría	Arquitectura Monolítica	Arquitectura de microservicios
Código	Código fuente único para toda la aplicación.	Código base independiente para cada microservicio.
Comprensibilidad	A menudo confuso y difícil de mantener.	Mayor facilidad de lectura y mantenimiento.
Despliegue	Implementaciones complejas con ventanas de mantenimiento y paradas programadas.	Despliegue sencillo, cada microservicio se puede implementar de forma individual, con un tiempo de inactividad mínimo.
Tecnologías	Típicamente desarrollado utilizando homogeneidad de pilas tecnológicas.	Cada microservicio es desarrollado utilizando heterogeneidad de tecnologías.
Escalamiento	Requiere escalar la aplicación entera	Cada microservicio puede ser escalado de manera independiente

Nota. Esta tabla muestra una comparativa de los factores relevantes dentro del desarrollo del software con un enfoque a las aplicaciones web de dos tipos de arquitecturas que se usan para dichos desarrollos, la arquitectura monolítica mvc y la arquitecta de microservicios. Recuperada de Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web (p5), por Daniel López, Edgar Maya. Séptima Conferencia de Directores de Tecnología de Información, TICAL 2017

Metodologías de desarrollo de software

Una metodología de desarrollo de software es un enfoque estructurado que se utiliza para planificar, organizar y controlar el proceso de desarrollo de sistemas de información(Maida & Pacienzia, 2015). Con el paso de los años, se han desarrollado diversas metodologías con sus propias fortalezas y debilidades. No todas las metodologías son adecuadas para todos los proyectos, ya que cada una es más adecuada para tipos específicos de proyectos basados en consideraciones técnicas, organizativas, de proyecto y de equipo.

Una metodología de desarrollo de software consiste en un marco de trabajo que incluye una filosofía de desarrollo de software y uno o varios enfoques del proceso de desarrollo. Además, se utilizan herramientas, modelos y métodos para facilitar el proceso de desarrollo de software. Estos marcos de trabajo suelen estar asociados con organizaciones encargadas del desarrollo, soporte y promoción de la metodología. A menudo, la metodología se documenta de forma formal(Maida & Pacienzia, 2015).

La metodología de desarrollo de software se utiliza para lograr un objetivo específico en un trabajo o proyecto informático. Consiste en elegir las técnicas adecuadas y los métodos correctos para completar las tareas de manera eficiente y efectiva con el objetivo de obtener una eficiente evolución del software en todas sus fases, incluyendo su puesta en marcha y mantenimiento(Maida & Pacienzia, 2015).

Software

Es el conjunto de elementos intangibles y lógicos que forman a un sistema informático, abarcando los componentes lógicos necesarios para realizar tareas en específico(Maida & Pacienzia, 2015). En términos más sencillos, se trata de todos los programas de computadora, normas, y documentación que importantes para el funcionamiento de un sistema informático. (Maida & Pacienzia, 2015) lo explican de esta manera en su investigación.

En la actualidad, el software juega un papel muy importante para la sociedad ya que impulsa la mayoría de las actividades y procesos en diversas áreas, incluyendo la industria, la educación, la salud, el entretenimiento y muchas otras.

Tipos de software

Existen diferentes tipos de software que se clasifican según su función, naturalidad y el propósito para el que fueron diseñados. Algunas de las categorías más comunes de software incluyen:

Software de sistema. Diseñado para proporcionar una plataforma y entorno de ejecución para otros programas y aplicaciones. Incluye el sistema operativo (software fundamental) que se encarga de la gestión del hardware y permite la ejecución de diferentes programas. Este tipo de software también incluye controladores de dispositivos, utilidades del sistema y software de gestión de redes.

Software de aplicación. Es el software diseñado para realizar tareas específicas y resolver necesidades concretas del usuario. Los programas de procesamiento de texto, hojas de cálculo, navegadores web, reproductores multimedia, software de diseño gráfico y herramientas de edición multimedia son algunos ejemplos de este tipo de software.

Software de programación. Este tipo de software está diseñado para ayudar a los programadores a desarrollar, depurar y mantener otros programas. Los entornos de desarrollo integrados IDE por sus siglas en inglés (Integrated Development Environment), compiladores, editores de texto avanzados y depuradores son ejemplos de este tipo de software.

Características

Los diferentes tipos de software comparten ciertas características básicas, tales como:

- Posee una funcionalidad específica y bien definida que determina su propósito principal.
- Posee una interfaz de usuario que permite la interacción con el programa.
- Es eficaz en el cumplimiento de su función y eficiente en términos de uso de recursos.

- Requiere de actualización y mantenimiento.
- Es seguro y estable.
- Se acompaña de documentación que describe su uso, instalación, configuración y detalla sus características y funcionalidades.
- Debe ser compatible con el hardware y otros programas con los que interactúa.
- Puede tener diferentes tipos de licencias y acuerdos de propiedad que regulan su uso y distribución.

Es importante tener en cuenta que, a pesar de que estas características se aplican para cualquier tipo de software, dichos tipos pueden tener características únicas y específicas que los distinguen dentro de sus respectivas categorías. Estas características específicas se ajustan a los propósitos y necesitades particulares de cada tipo de software.

Metodologías tradicionales

Las metodologías tradicionales en algunas ocasiones son llamadas "pesadas" porque se centran en realizar una documentación exhaustiva y controlada del proyecto desde el principio, con especificaciones precisas y un plan de trabajo definido. Estas metodologías conciben al proyecto como uno solo e imponen una disciplina rigurosa en el proceso de desarrollo del software, lo que puede hacerlo más eficiente. Sin embargo, son inflexibles ante cambios de requisitos y pueden resultar costosas, y a la vez son poco adaptables en entornos cambiantes. Esto contrasta con las metodologías ágiles, que son más flexibles y se adaptan mejor a los cambios.

Metodologías Ágiles

Este enfoque se presenta como una alternativa a los problemas que pueden surgir con las metodologías tradicionales y se basa en dos aspectos fundamentales, retardar las decisiones y la planificación adaptativa. Su fundamento se basa en la capacidad de adaptación de los procesos de desarrollo.

Un modelo de desarrollo ágil generalmente consiste en un proceso incremental (entregas frecuentes con ciclos rápidos), cooperativo (trabajo constante y estrecha comunicación entre clientes y desarrolladores), sencillo (método fácil de aprender y modificar para el equipo) y adaptable (capaz de permitir cambios de último momento). Las metodologías ágiles ofrecen directrices, principios y enfoques prácticos simplifican la ejecución de proyectos y experiencias positivas tanto para clientes como para equipos de colaboradores(Maida & Pacienzia, 2015). Estas metodologías enfatizan que la capacidad de respuesta a los cambios es más importante que seguir un plan estricto; presentando como principal particularidad la flexibilidad.

SCRUM. Es una metodología ágil para desarrollo de software, se fundamenta en un marco de trabajo creados para fomentar la cooperación eficiente entre equipos dentro de un proyecto. Su objetivo principal es permitir la entrega de productos de alta calidad de manera rápida y flexible. Scrum hace uso de una estrategia gradual y se apoya en la teoría de control empírico de procedimientos, la cual se sustenta en la transparencia, la inspección y la adaptación.

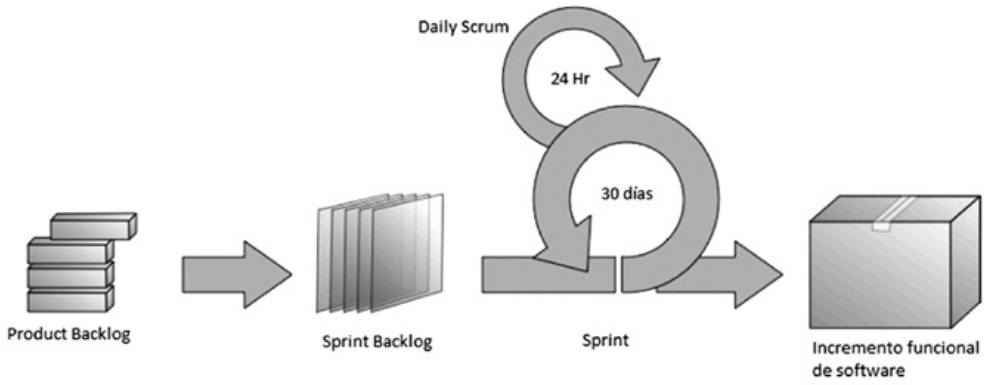
Emplea un enfoque interdisciplinario y se divide el proyecto en múltiples fases que se traslapan entre sí(Cadavid et al., 2013). Los llamados equipos Scrum son autogestionados, multifuncionales y trabajan en iteraciones(Patricia et al., 2018). La autogestión les permite elegir la mejor forma de realizar el trabajo, en lugar de seguir lineamientos de personas externas al equipo. Además, los integrantes del equipo tienen todos los conocimientos necesarios para llevar a cabo el trabajo, lo que los hace multifuncionales(Cadavid et al., 2013).

Esta metodología define tres roles principales. El Scrum Master tiene la función de asegurar que el equipo adopte la metodología, sus prácticas, valores y normas, pero no gestiona el desarrollo en sí. El dueño del producto es responsable de definir, actualizar y ordenar los requerimientos del producto en una lista llamada Product Backlog(Cadavid et al.,

2013). Por su parte, el equipo de desarrollo se encarga de implementar las funcionalidades y entregar el producto en iteraciones. Además, utiliza varios artefactos que proporcionan dirección y transparencia al equipo. Entre estos artefactos se encuentran el Product Backlog, que es una lista de los requerimientos ordenados por valor, riesgo, prioridad y necesidad; el Sprint Backlog, que es la lista de tareas que el equipo selecciona para trabajar durante un período de tiempo llamado Sprint; el Monitoreo de Progreso, que permite visualizar el avance del proyecto; y el incremento, que es la versión del producto que se entrega al final de cada iteración.

Figura 2

Fases de un Sprint



Nota. El gráfico representa las fases de un sprint y como cada una de estas hace uso de los artefactos utilizados por la metodología SCRUM para aportar dirección y transparencia en los equipos de trabajo. Tomado de Revisión de metodologías ágiles para el desarrollo de software (p.33), por Cadavid, Andrés Navarro, Martínez, Juan Daniel Fernández y Vélez, Jonathan Morales, 2013, Prospectiva

Figma. Es una plataforma en línea para la creación de interfaces de usuario que ha transformado la manera en que los equipos de diseño colaboran y crean experiencias visuales interactivas. En el ámbito del diseño de software y desarrollo de aplicaciones, Figma desempeña un papel esencial al proporcionar un entorno de diseño integral que permite a los diseñadores crear, compartir y colaborar en proyectos de manera eficiente.

Postman. Es una herramienta esencial para el desarrollo y pruebas de APIs, ampliamente utilizada en la creación y mantenimiento de servicios web. Como cliente API, Postman permite a los desarrolladores diseñar, enviar y recibir solicitudes HTTP y evaluar las respuestas obtenidas. Además de simplificar la interacción con APIs, Postman facilita la construcción de solicitudes complejas, el ajuste de parámetros y la visualización detallada de los resultados. A través de su interfaz intuitiva y su capacidad para automatizar pruebas, Postman agiliza el proceso de desarrollo, depuración y documentación de APIs, lo que contribuye a garantizar la calidad y eficiencia en la comunicación entre aplicaciones(API Documentation Tool | Postman, s/f).

Railway. es una plataforma que busca simplificar y acelerar el proceso de desarrollo y despliegue de aplicaciones web y APIs. Ofrece una interfaz visual para construir, desplegar y gestionar proyectos en la nube, eliminando gran parte de la complejidad asociada con la infraestructura y configuración. Railway integra funciones de CI/CD (Integración Continua / Entrega Continua), automatizando la construcción, prueba y despliegue de aplicaciones. A través de su enfoque basado en contenedores, Railway permite a los desarrolladores concentrarse en el código y la funcionalidad, sin preocuparse por la infraestructura subyacente. Con su capacidad para adaptarse a diversas pilas tecnológicas, Railway agiliza la creación de aplicaciones y servicios en la nube, ofreciendo una alternativa versátil y eficiente para equipos de desarrollo(Railway Docs, s/f).

Metodologías Tradicionales vs Ágiles

En el desarrollo de proyectos de software, la elección de la metodología adecuada es un factor determinante para el éxito del proyecto. A lo largo de los años, han surgido dos enfoques principales para la gestión y desarrollo de proyectos: las metodologías ágiles y las tradicionales. Cada enfoque tiene sus propias características, ventajas y desafíos, lo que hace

que la elección entre ambos sea una decisión crucial para los equipos de desarrollo y las organizaciones.

Las metodologías tradicionales, como el modelo en cascada, han sido ampliamente utilizadas durante décadas y se caracterizan por su enfoque secuencial y planificado. Estas metodologías buscan una planificación detallada desde el inicio del proyecto, estableciendo una estructura rígida y bien definida para el desarrollo del software. Por otro lado, las metodologías ágiles, como Scrum, han ganado popularidad en los últimos años debido a su enfoque iterativo e incremental, que se adapta mejor a proyectos en constante cambio y con requisitos poco claros o cambiantes. Estas metodologías promueven la flexibilidad, la colaboración y la entrega de valor de forma rápida y continua.

Tabla 4

Metodologías tradicionales vs metodologías ágiles

Metodologías tradicionales	Metodologías ágiles
Predictivos	Adaptativos
Orientados a procesos	Orientados a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdivido en proyectos pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de software al finalizar el desarrollo	Entregas de software constantes
Documentación extensa	Poca documentación

Nota. Esta tabla muestra los aspectos más relevantes de las metodologías de desarrollo tradicional contrastándolas con las metodologías ágiles. Tomado de Revisión de metodologías ágiles para el desarrollo de software (p.33), por Cadavid, Andrés Navarro, Martínez, Juan Daniel Fernández y Vélez, Jonathan Morales, 2013, Prospectiva

Capítulo III: Desarrollo

En este capítulo se describe el desarrollo del prototipo del proyecto. Se incluye el proceso de selección de la arquitectura, la metodología de desarrollo de software implementada y las herramientas utilizadas de software y hardware.

Requerimientos Funcionales

El prototipo debe cumplir tres funcionalidades principales. La primera funcionalidad tiene por objetivo presentar al usuario la información en tiempo real obtenida del nodo de sensores, el cual se encuentra conectado al Gateway TTN y desplegada en una aplicación dentro de la nube de The Things Network, esta aplicación envía la información mediante un protocolo de comunicación hacia los microservicios los cuales son consumidos por el dashboard web, permitiendo al usuario obtener lecturas de los sensores en tiempo real. La segunda funcionalidad del prototipo es almacenar las lecturas realizadas por el nodo en una base de datos MongoDB en la nube, de tal manera que pueda ser visualizada en el dashboard en forma de cuadros comparativos. Finalmente, la tercera funcionalidad consta de transmitir en tiempo real el video obtenido de la emulación de un dron a través de una cámara conectada a un dispositivo ESP32 y a un Arduino UNO.

Arquitectura del prototipo

Como se ha descrito en el capítulo anterior algunas de las arquitecturas de software pueden ser seleccionadas para el desarrollo del proyecto; sin embargo, de acuerdo a los requerimientos funcionales y tomando como referencia lo mencionado por Tatiana Gómez Suárez (et al., 2018) el enfoque de una arquitectura basada en microservicios posee ventajas a nivel de escalabilidad horizontal, y uso constante de recursos de red y comunicaciones entre los distintos elementos que conforman la arquitectura así como la flexibilidad y escalabilidad que esta arquitectura brinda. Debido a los factores previamente mencionados se decidió utilizar

la arquitectura basada en microservicios para el desarrollo del proyecto con la finalidad de aprovechar las ventajas de esta arquitectura.

Metodología de desarrollo de software

La metodología de desarrollo de software que se escogió para el desarrollo del prototipo fue la metodología SCRUM, ya que es un framework de desarrollo de software de metodología ágil el cual permite se basa en un enfoque incremental y se fundamenta en la teoría de control empírico de procesos (Ozerienska,2016). SCRUM cubre la necesidad de desarrollo continuo y simultáneo de los microservicios de acceso a datos, como el desarrollo de la interfaz de usuario.

Selección de tecnologías

Frameworks

Espinoza (2021) menciona en su análisis los frameworks más utilizados en el desarrollo de aplicaciones web como se puede apreciar en la Tabla 5.

Tabla 5

Frameworks más utilizados en el año 2020

N°	Framework
1	Laravel
2	Django
3	VueJS
4	React
5	Ruby on Rails
6	Node.js
7	Angular

Nota. Esta tabla muestra un extracto de los framework más utilizados en el año 2020. Recuperado de Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web, Raquel Espinoza Hurtado, CEDEMAZ

Para llevar a cabo el proyecto, se optó emplear el framework Angular para el desarrollo del frontend, partiendo como base del proceso de elección la Tabla 5, así como las características de integración que soporta este framework, permitiendo la creación de componentes reutilizables y la gestión eficiente de dependencias, la incorporación de un sistema de enrutamiento que permite la navegación entre diferentes vistas y la creación de aplicaciones de una sola página (SPA) de manera sencilla (Google, 2023). Además, permite crear directivas personalizadas que extienden el comportamiento del DOM y brinda una manera flexible de manipular el contenido del sitio web. En cuanto a pruebas, angular incorpora herramientas y soporte para escribir pruebas unitarias y de integración, lo que facilita la creación de código robusto y confiable.

Para el desarrollo del backend se consideró lo mencionado Armash Aslam (et al., 2015), en el cual recomienda python apoya al desarrollo web a través del microframework Flask, ya que este proporciona la funcionalidad básica de un framework web y permite añadir más librerías para que la funcionalidad y el conjunto de características se puedan ampliar a un nuevo nivel. Por estas razones se ha decidido escoger flask como parte de la tecnología de desarrollo de los microservicios.

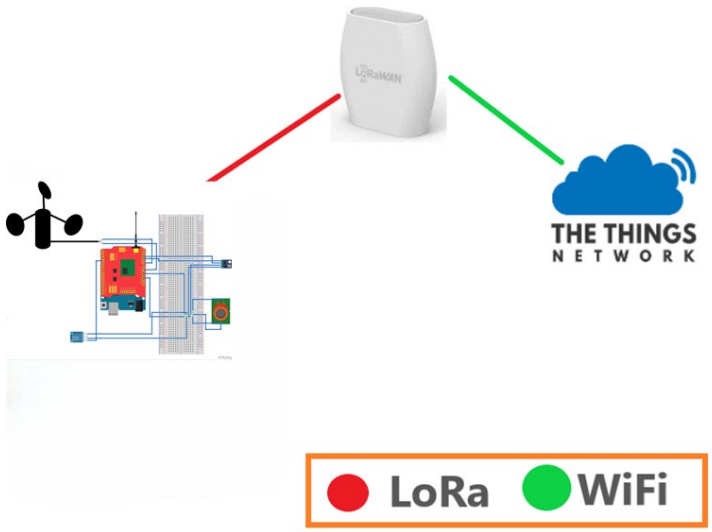
Herramientas de hardware

Los dispositivos de hardware utilizados en el desarrollo del prototipo fueron determinados por la arquitectura propuesta en el sistema de información para alertas de incendios forestales, tales como:

- Nodo o módulo de recolección de información
- Indoor Gateway TTN.
- Arduino UNO
- ESP32-CAM

Figura 3

Diagrama general de la arquitectura propuesta por el sistema de alerta de incendios forestales



Nota. Esta figura representa de manera general la arquitectura del sistema de información de alerta de incendios forestales, en la misma se representa el Indoor Gateway que permite la comunicación con el servidor de red LoRaWan.

Herramientas de software

Dado que en el capítulo anterior se muestran los diferentes tipos de software existentes, es necesario determinar los tipos que serán seleccionados para el desarrollo del prototipo del dashboard.

Software de programación

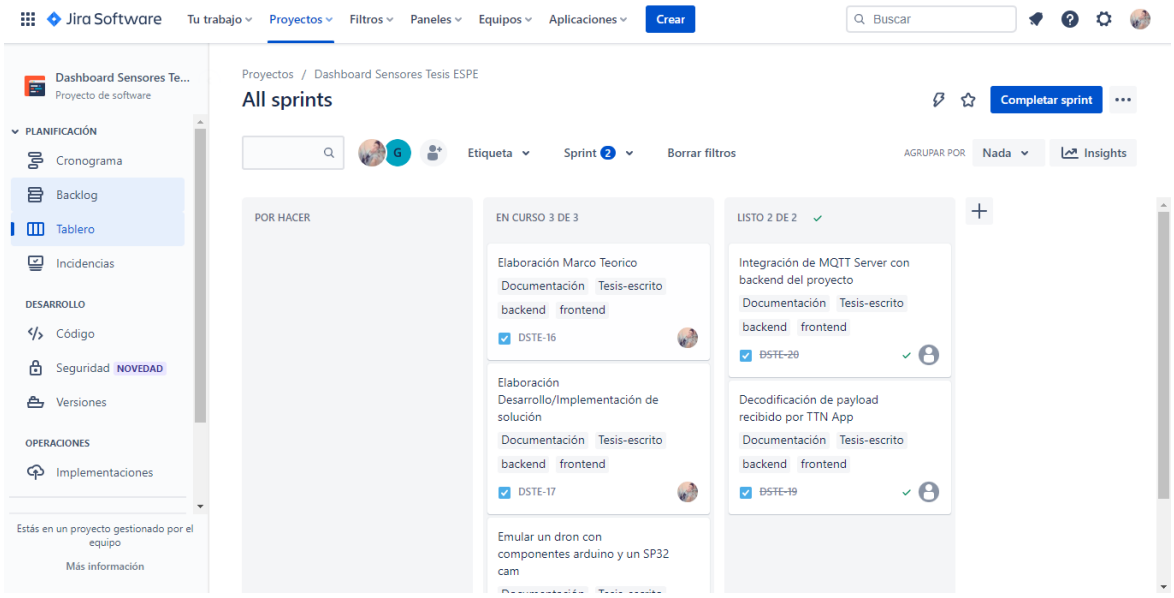
Para la construcción del prototipo, se previó utilizar IDEs de programación como son WebStorm y PyCharm, los cuales son softwares de programación desarrollados por la compañía Jetbrains y diseñados específicamente para interpretar lenguajes de programación como TypeScript y Python respectivamente.

Software de control del proceso de desarrollo.

Dentro de esta clasificación se hizo uso de software como JIRA Software, con el enfoque de gestionar el proyecto y poder aplicar la metodología SCRUM; al mismo tiempo, este se complementó con software de aplicación como Git que nos permitió controlar el versionamiento del código tanto para backend como para frontend. Esto permitió integrar las diferentes ramas del proyecto con JIRA para darles el debido seguimiento y poder identificar incidencias del proyecto y no solo a nivel de código.

Figura 4

Aplicación de Metodología SCRUM usando JIRA Software



Nota. La figura representa el uso de JIRA Software como software de control, gestión y seguimiento del desarrollo del proyecto "Prototipo de un dashboard para el sistema de información de incendios forestales".

Software de administración de base de datos NoSQL

Otro software de aplicación que se utilizó en el desarrollo del prototipo es MongoDB Compass, la interfaz de usuario que permite gestionar bases de datos MongoDB.

IaaS y PaaS

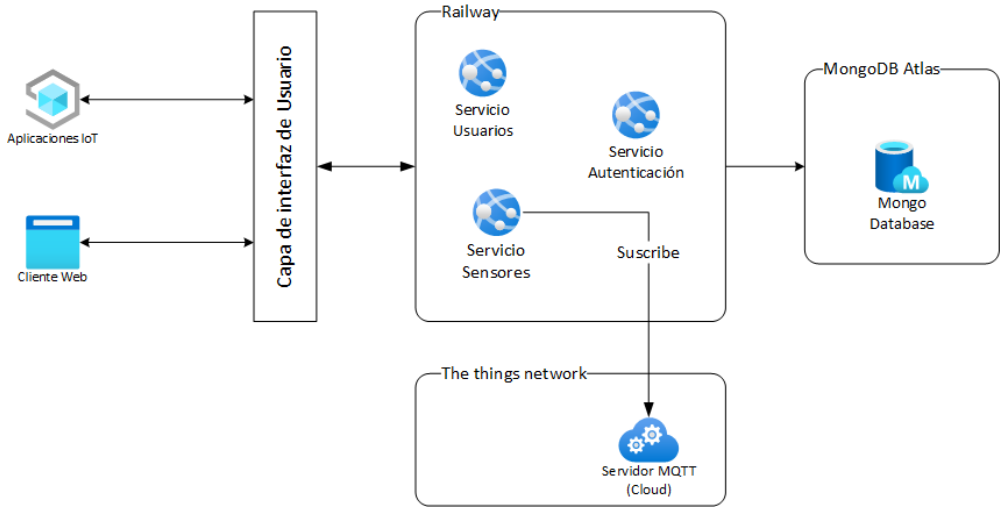
Ambas tecnologías de computación en la nube fueron utilizadas por parte de Thing Stack y su servidor de red LoRaWan, el mismo provee diferentes tipos de servidores y clústeres en la nube que permiten la gestión de aplicaciones, dispositivos finales y Gateways.

Arquitectura propuesta para la solución

Tras la identificación de todos los componentes que forman parte de la solución se planteó la siguiente arquitectura como propuesta para el desarrollo de la solución del trabajo de investigación.

Figura 5

Diagrama de arquitectura propuesta



Nota. Esta figura representa la arquitectura propuesta para el desarrollo de la solución

Planteamiento de la solución

Tras el análisis y la selección de herramientas adecuadas para el óptimo avance del proyecto, se llevó a cabo la asignación de actividades específicas para cada miembro del equipo. Una vez seleccionada la metodología de desarrollo y con base en los requerimientos funcionales, se procedió a la selección y modelado de la base de datos. Al analizar la

información que el nodo proporcionaba al Gateway los cuales se apreciaron como datos no estructurados, se decidió utilizar una base de datos NoSQL, debido a la flexibilidad a nivel de integridad referencial que posee este tipo de base de datos, considerando el factor de la posibilidad de agregar sensores al nodo en futuras versiones del proyecto.

Implementación del nodo de sensores y enlace al Gateway TTN

Como se describe en el capítulo II, el presente proyecto de desarrollo fue basado en el trabajo propuesto por Orellana (et al., 2021), en dicho trabajo se implementó un nodo de sensores mediante Arduino UNO, sensores de lectura de: temperatura, presión barométrica, luminosidad. Por lo cual, se realizó el ensamblaje del nodo basado en el mismo diseño, y se procedió a cargar el archivo de lectura de sensores en el Arduino UNO. Posteriormente se observó la información del monitor del IDE del Arduino para comprobar el correcto funcionamiento del nodo de sensores.

Como se mencionó con anterioridad, el nodo realiza el envío de la información recopilada hacia el servidor de red LoRaWan en The Thing Stack mediante el Gateway, el mismo decodifica la información y la envía hacia The Thing Stack usando el protocolo MQTT.

El protocolo MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería ligero y eficiente diseñado para facilitar la comunicación entre dispositivos conectados a Internet de las Cosas (IoT). MQTT sigue un enfoque de publicación/suscripción, donde los dispositivos pueden publicar mensajes en ciertos temas (topics) y suscribirse a temas específicos para recibir mensajes. Es altamente escalable y está diseñado para ser utilizado en entornos con ancho de banda limitado y conexiones inestables, lo que lo hace ideal para aplicaciones IoT donde se requiere una comunicación confiable y con baja sobrecarga de red. Los mensajes enviados a través de MQTT son muy livianos, lo que minimiza la sobrecarga de la red y lo hace especialmente adecuado para dispositivos con recursos limitados, como

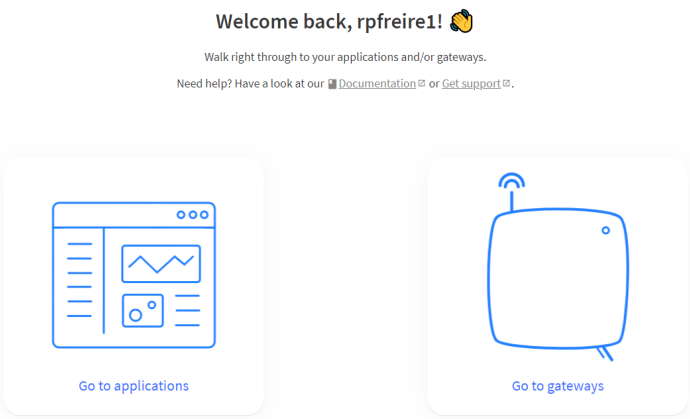
sensores y actuadores en el contexto de aplicaciones de monitoreo y prevención de incendios forestales.

Esto quiere decir que, para poder acceder a la información y almacenarla en nuestra base de datos, debemos suscribirnos a los mensajes que son enviados a nuestro servidor en The Thing Stack. Para esto debemos registrarnos en su sitio oficial y, una vez que obtengamos una cuenta, se podrá acceder a la consola de administración para crear aplicaciones y agregar dispositivos finales o gateways.

Para poder acceder a una integración con servidor MQTT por parte de The Things Stack, primero debemos registrar nuestro gateway, desde la pantalla de inicio se debe escoger la opción “Go to gateways” para poder agregar nuestro dispositivo Gateway.

Figura 6

Registro del Gateway en la consola de la nube TTN.



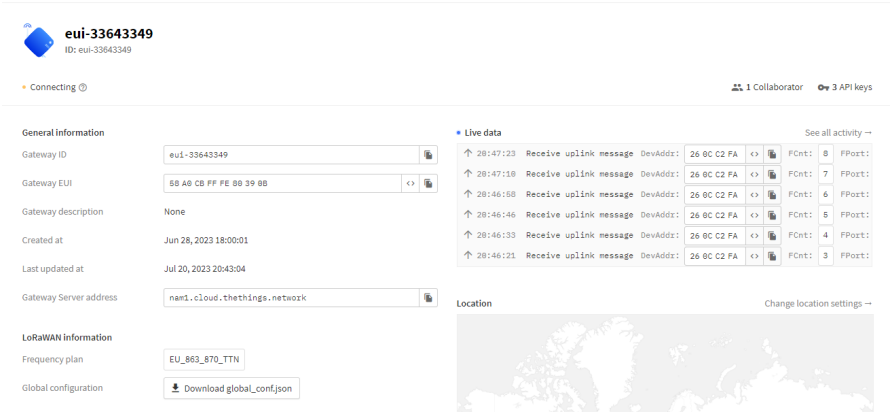
Nota. La figura el registro del Gateway en la consola de la nube de The Things Network TTN.

Para agregarlo se debe recurrir a la documentación proporcionada por el sitio oficial de The Things Stack. Una vez registrado nuestro dispositivo Gateway, lo podremos visualizar en la pantalla de gateways, aquí es dónde la consola nos permite acceder a nuestro dispositivo y visualizar todo el tráfico que recibe y envía.

Además de que la información es no estructurada una base de datos MongoDB es la más adecuada por la adaptabilidad a este tipo de datos, sin mencionar la escalabilidad y flexibilidad que este tipo de base de datos presenta y nos permitirá manejar grandes volúmenes de datos.

Figura 7

Gateway Funcionando



Nota. La figura representa el dispositivo Gateway en funcionamiento desde la consola de The Things Stack, siendo el caso de que se haya seguido de manera correcta la documentación proporcionada

Con el Gateway en correcto funcionamiento debemos considerar el dato que nuestra información en vivo nos devuelve, la cual indica la dirección del equipo o DevAddr. Esto es de suma importancia ya que nos servirá para configurar la aplicación dentro de The Things Stack.

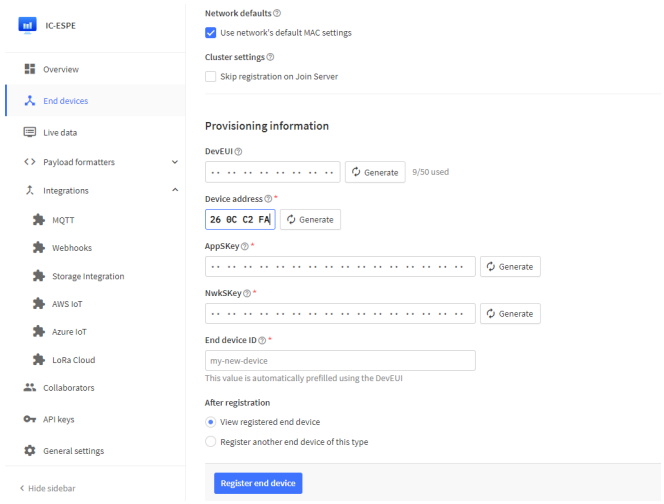
Continuando con la creación y configuración del aplicativo, debemos dirigirnos a nuestra página de resumen y ahora se debe seleccionar la opción “Go to applications”. Con esta opción la consola nos permitirá crear una nueva aplicación y configurar nuestra dirección de dispositivo, de igual manera se debe acceder a la documentación en donde se detalla a profundidad cómo añadir una aplicación a The Thing Stack.

Una vez creada la aplicación, se debe registrar el dispositivo final de manera manual, para esto necesitamos del parámetro DevAddr que nos devuelvo el Gateway y procedemos a

llenar la información del formulario. Cabe mencionar que las únicas opciones que vamos a generar automáticamente son DevEUI, AppSKey y NwkSkey. El id del end device es un campo de texto cualquiera, y se lo nombra de la manera que se crea necesaria.

Figura 8

Formulario de registro de dispositivo final

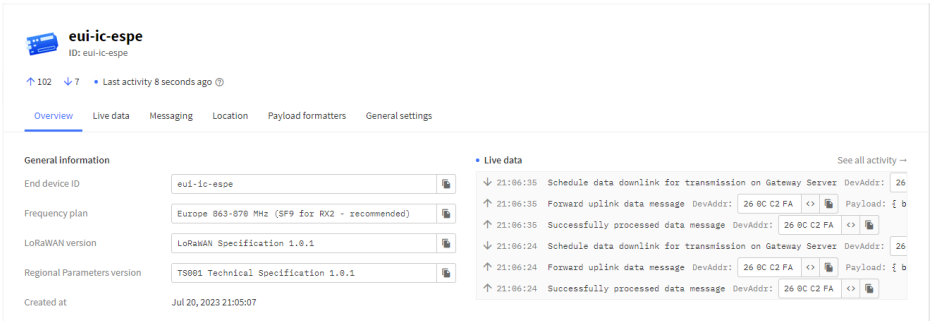


Nota. La figura representa la manera en la que se debe llenar el formulario de registro de un dispositivo final en la consola de TTN

Si el registro se efectuó correctamente, lo que deberíamos ver a continuación es nuestro dispositivo final agregado al aplicativo devolviendo información en vivo.

Figura 9

Dispositivo final configurado en TTN



Nota. La figura representa el dispositivo final configurado correctamente en la consola de TTN

Configuración de la conexión MQTT entre el nodo y el gateway

A continuación, se debe añadir las integraciones, así como la decodificación de los payloads recibidos y enviados por el aplicativo; para esto empezaremos por añadir la integración MQTT. En nuestra página de resumen contaremos con un menú lateral que contempla la opción “Integrations” y dentro de ella una opción “MQTT”, al dar clic en aquella opción, The Things Stack despliega los parámetros de configuración para acceder al servidor MQTT y poder suscribirnos a los mensajes de subida y de descarga. Cabe mencionar que TTN proporciona un endpoint seguro usando TLS y un endpoint público.

Una vez anotada las credenciales de conexión procederemos a decodificar nuestro payload, en el contexto de tecnologías de la información, incluyendo redes, comunicaciones inalámbricas, sistemas de almacenamiento de datos y protocolos de transmisión. Payload se refiere a la información real que se está enviando o recibiendo de cualquier otro tipo de información relacionada con la transmisión o el control. Para esto TTN nos provee la opción “Payload formatters” en su menú lateral, dentro de la cual encontraremos la opción “Uplink”, dicha opción nos permitirá elegir qué tipo de formateador necesitamos; la documentación de TTN describe cada uno de los tipos que se pueden elegir. Sin embargo, para el desarrollo del

prototipo hemos elegido CayenneLPP, debido a que utiliza la decodificación del payload en base a los API Key que se generan del aplicativo y del servidor MQTT.

Desarrollo del Backend con Flask

En esta sección se profundizará en el proceso de desarrollo del backend, utilizando Flask como el framework elegido. Se describirán los pasos clave involucrados en la creación de rutas y controladores para recibir y procesar datos del nodo de sensores, así como la integración de la API de TTN para obtener los datos. Además, se explicará cómo se logra el almacenamiento efectivo de los datos en la base de datos MongoDB.

Flask como framework para el backend

Flask, un microframework de Python, se utiliza como la base para desarrollar el backend del sistema. Su diseño minimalista y modular permite crear aplicaciones web de manera rápida y eficiente, al tiempo que brinda flexibilidad para incorporar características específicas del proyecto. Flask utiliza el concepto de rutas y controladores para gestionar las solicitudes HTTP entrantes y determinar cómo deben procesarse y responderse.

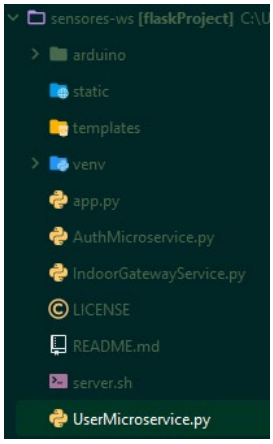
Creación de microservicios para la gestión del lado del servidor

Es conocido por su simplicidad y flexibilidad, lo que lo hace ideal para desarrolladores que desean construir aplicaciones web sin la complejidad y restricciones de otros frameworks más grandes. Es conocido como un "microframework" debido a su enfoque mínimo y su tamaño compacto, lo que lo hace fácil de aprender y usar, incluso para aquellos que recién comienzan en el desarrollo web con Python. A pesar de su simplicidad, Flask es lo suficientemente potente para manejar aplicaciones web más complejas, y muchos desarrolladores aprecian su enfoque minimalista y su capacidad para mantener el código limpio y legible.

Para el desarrollo de los microservicios se utilizó una versión de Python 3.11, y el IDE PyCharm; la descripción de la funcionalidad de cada microservicio se detalla a continuación.

Figura 10

Microservicios creados con flask framework



Nota. La figura representa los diferentes microservicios creados usando el microframework flask y el IDE PyCharm, así como la estructura del directorio de trabajo del proyecto

UserMicroservice. Utiliza diferentes librerías de flask, como jsonify que permite el formateo de una respuesta hacia el frontend en formato JSON; PyMongo, librería que permite integrar al microservicio funcionalidades esenciales para la conexión de la base de datos y tratamiento de datos. El servicio expone las APIs necesarias para la gestión de usuarios, permite consular la información de usuarios por Id, crear usuarios y eliminar usuarios.

AuthMicroservice. El servicio utiliza librerías de flask mencionadas en el microservicio de usuarios para el envío de respuestas, conectividad con la base de datos; adicional a eso, implementa librerías de autenticación para transmitir información de manera segura entre dos partes, permitiendo al usuario autenticarse y compartir la información de manera secreta y segura entre el servicio API y la aplicación.

SensorsMicroService. El servicio implementa una librería en especial que se llama paho.mqtt.client, la misma permite configurar el acceso al servidor MQTT con las credenciales

que TTN proporcionó al añadir la integración a la aplicación. Adicional a esto la librería permite suscribirse al servidor y escuchar todos los mensajes con el tema (topic) que TTN describe en la documentación, dicho tema permitirá suscribirse a todos los mensajes de carga usando el tema v3/sensores-espe@ttn/devices/eui-ics23/up.

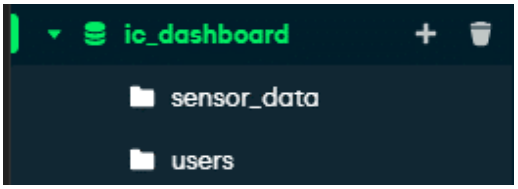
Como resultado, el servicio devuelve todas las APIs que exponen los datos de humedad relativa, temperatura, iluminación y presión barométrica, tanto para consultas de información almacenada, como la información en tiempo real.

Almacenamiento de los datos en MongoDB

Continuando con el modelado de la base de datos, este integra el Backend del aplicativo, es decir, se encargará de satisfacer la gestión y autenticación de usuarios, así como de permitir manejar el histórico del nodo. Para esto se creó una base de datos y dos colecciones.

Figura 11

Modelo de la base de datos mongo



Nota. La figura describe la base de datos y las colecciones que se guardan en ella para la integración del backend del prototipo

Diseño y Desarrollo del Dashboard con Angular

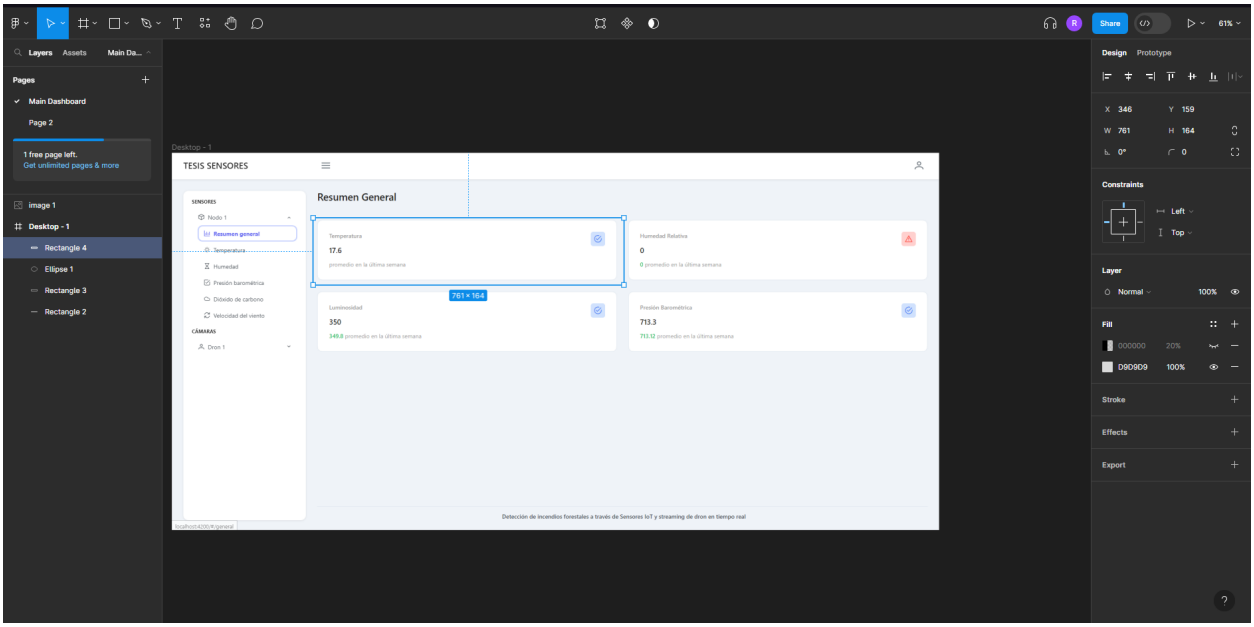
La creación de un dashboard efectivo es esencial para proporcionar a los usuarios una interfaz intuitiva y atractiva para visualizar los datos recopilados. En esta sección se detalla el proceso de diseño y desarrollo del dashboard utilizando el framework Angular, destacando cómo se logra la presentación visual de los datos de manera comprensible y significativa.

Diseño de la estructura del dashboard y sus componentes

Para el desarrollo del diseño de prototipos se utilizó Figma como se puede observar en la Figura 11, esto permitió comprender de mejor manera el flujo de la información, así como determinar requerimientos no funcionales que no fueron considerados en la aproximación inicial, tales como el planteamiento de sesiones en las vistas e inicio de sesión.

Figura 12

Diseño de prototipos del dashboard en Figma.



Nota. La figura describe el diseño del prototipo, la estructura del dashboard y las opciones contenidas en la barra de navegación.

Existen dos fuentes de información que alimentan la interfaz gráfica del usuario, la primera es el nodo con sus sensores, y el segundo es el dron, el cual transmite vía streaming directamente al aplicativo front-end.

Estas fuentes de información se encuentran estructuradas mediante tres módulos principales: el módulo de información general, el cual despliega la información en tiempo real de la última lectura obtenida por el nodo de sensores, y el módulo de información de cada uno

de los sensores, el cual muestra mediante gráficos estadísticos, la información recolectada de los nodos en el tiempo, la cual se encuentra almacenada en la base de datos de la nube. El módulo de dron muestra vía streaming en tiempo real la cámara de la simulación del dron.

Dentro de cada sensor del nodo se encuentran dos pestañas, una de información general la cual muestra información en tiempo real del último registro obtenido del sensor a través del Gateway, dentro de esta pestaña existen valores críticos los cuales se muestran con un ícono de alerta y un cambio de color en el indicador.

La segunda pestaña muestra una escala de valores del sensor vs el tiempo obtenido en días. Dentro de dicha escala existe un valor límite el cuál si sobrepasa la medición, diferenciará para toma de decisiones y análisis de dichos resultados.

Para el módulo de Dron, se realizó una simulación del servicio de streaming de un Dron, a través del Arduino Uno y un módulo ESP32 con una cámara. Este módulo transmite a través de una red inalámbrica Wifi, mediante una dirección IP asignada. La aplicación Front-End accede a esa dirección IP y a través de la librería angular-vlc y se puede configurar un componente el cual muestre el video transmitido por el módulo ESP32.

Figura 13

Estructura de la visualización de la información del dashboard.

Nota. La figura describe la estructura de la visualización del dashboard. Los módulos que se despliegan y las pestañas que se derivan de estos.

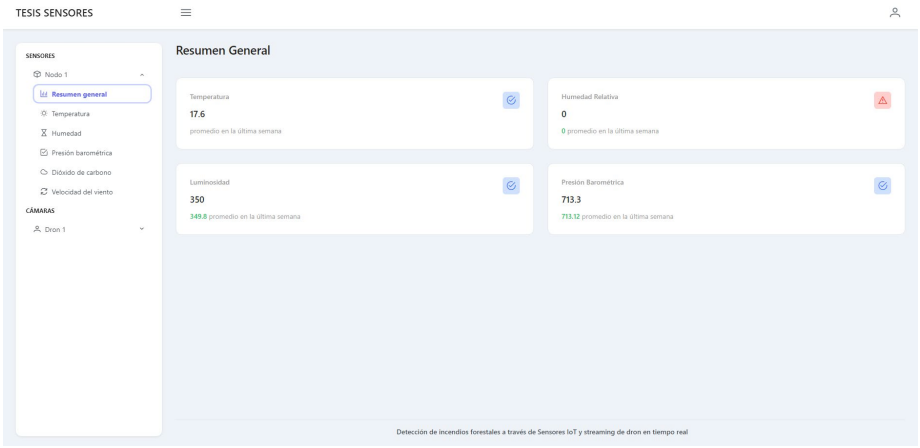
Módulo de información general. En este módulo se obtiene la información en tiempo real de la última lectura obtenida del nodo por el servidor TTN a través del backend. Esta información es actualizada cada cierto tiempo haciendo uso de un algoritmo enfocado a eventos, el cual recibe llamadas al backend en el momento que se realice una nueva lectura. Para la implementación del nodo de sensores se consumen los servicios Rest del backend a través de servicios creados en el front-end. El cual muestra tanto la información obtenida en tiempo real, como la información recopilada en la base de datos en forma de un gráfico vs tiempo.

El módulo de información general despliega la información de acuerdo con las respectivas unidades de medida de los sensores, y posee parámetros de lectura los cuales permiten identificar mediciones anómalas. Estas mediciones anómalas se pueden dar en dos sentidos:

- Lectura de estado 0. Esta lectura notifica que existe alguna falla en el sensor, error de lectura, error de conexión o el sensor se encuentra inhabilitado.
- Valor de lectura mayor al parámetro. Esta lectura notifica una lectura mayor a la establecida como normal en el parámetro, mediante este indicador se pueden identificar valores críticos de riesgo, los cuales pueden implicar un posible incendio forestal.

Figura 14

Visualización del módulo de información general con lecturas en tiempo real.



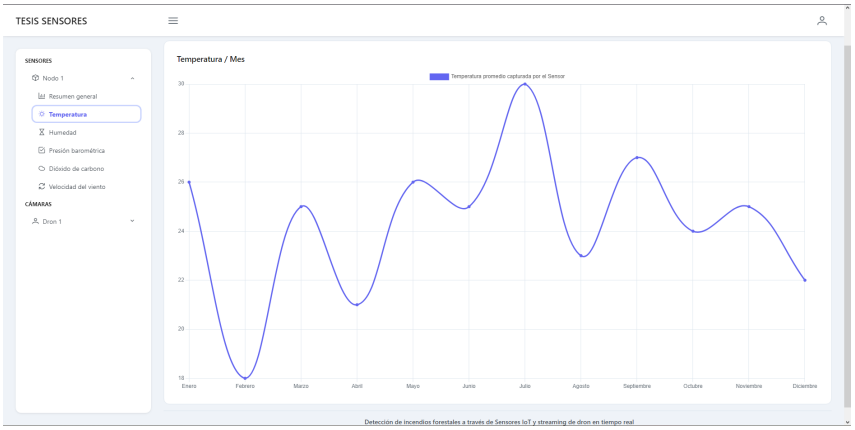
Nota. La figura describe el módulo de información general con el título Resumen General, desplegando la información obtenida en tiempo real del nodo.

Módulo de información de los sensores. Cada sensor cuenta con un módulo de información, el cual utiliza gráficos estadísticos con los valores de lectura obtenidos durante el tiempo. Estos gráficos estadísticos permiten observar los cambios de las mediciones en función del tiempo, lo cual permite tomar medidas predictivas frente a posibles escenarios de riesgos.

Los gráficos informativos poseen parámetros configurables de tiempo, lo cual permite al usuario obtener información con distintos parámetros de tiempo.

Figura 15

Visualización del módulo de información de los sensores con la gráfica de mediciones por tiempo.

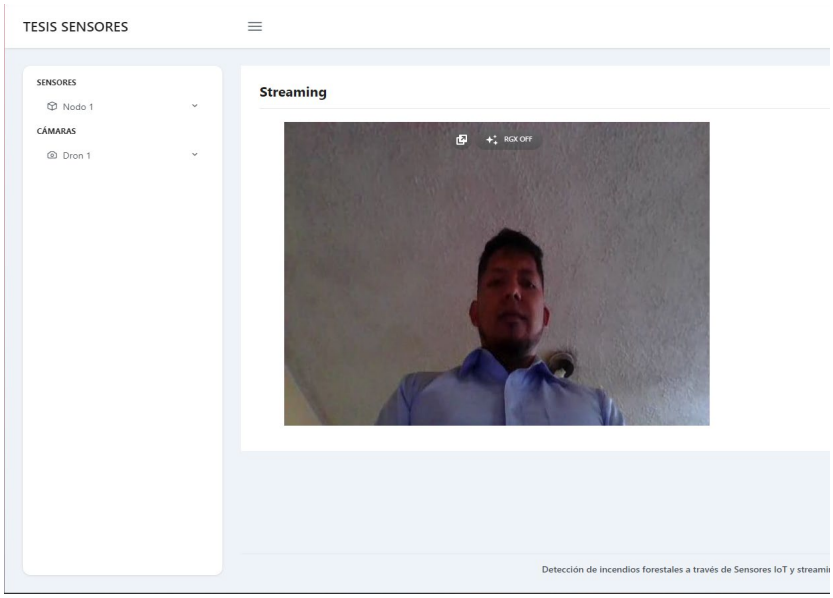


Nota. La figura describe el módulo de información del sensor de temperatura con el registro de mediciones de temperatura por tiempo obtenidos de la base de datos MongoDB en la nube Atlas.

Módulo de streaming de imagen de emulación de dron. Mediante una implementación de ESP32-CAM con FTDI, se realizó una emulación de streaming de video en tiempo real de la cámara, la cual mediante una dirección IP se comunica con el dashboard, el cual muestra el video en tiempo real.

Figura 16

Visualización del módulo de streaming con el video del simulador de dron.



Nota. La figura describe la visualización del módulo de streaming consumiendo el video del simulador del dron con Arduino UNO y ESP32-CAM.

Capitulo IV: Resultados obtenidos

En el marco de este estudio dedicado a la creación y desarrollo de un sistema de monitoreo y prevención de incendios forestales a través de un dashboard integrado en la nube, el capítulo IV presenta los resultados concretos y las realizaciones alcanzadas durante el proceso de implementación y evaluación. Este capítulo no solo destaca el logro de los objetivos planteados, sino que también proporciona una visión detallada de cómo las distintas etapas del proyecto han convergido para dar forma a un prototipo funcional y prometedor.

Conectividad del sistema de lectura de sensores y almacenamiento en la nube

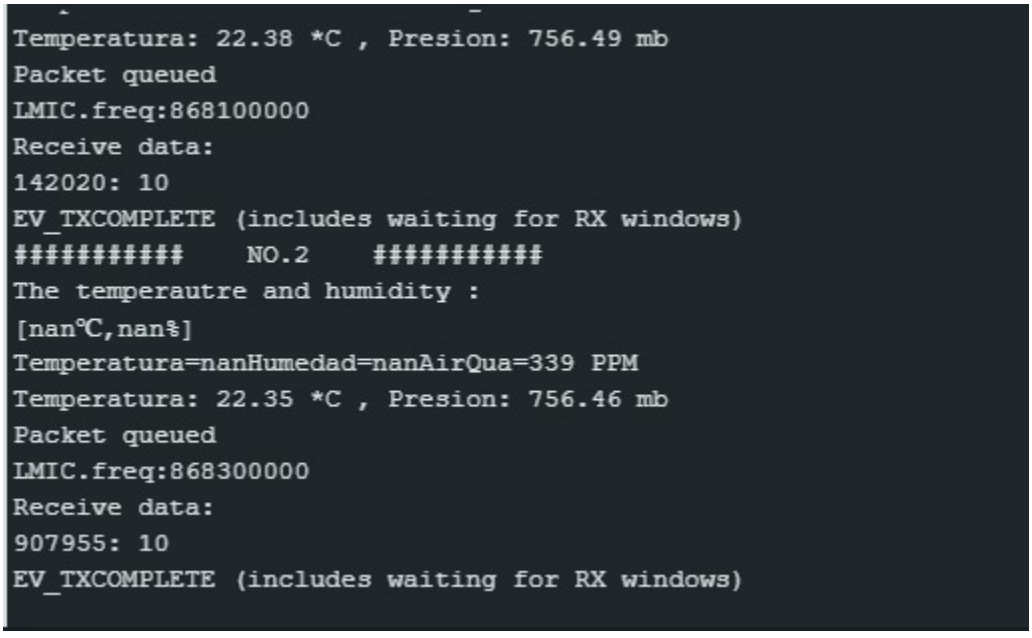
Una de las primeras evaluaciones realizadas al prototipo fue la evaluación de la conectividad de los distintos módulos que conforman dicho prototipo. Con el objetivo de comprobar la recopilación de la información del nodo, el tratamiento de la misma por el servidor IoT, la recepción de esta información a través de los microservicios del lado del servidor, para

finalmente ser desplegada al usuario a través de la interfaz gráfica tipo dashboard web.

La primera evaluación realizada fue la prueba de recolección de datos del nodo, mediante el uso del serial monitor dentro del Arduino IDE se pudo apreciar la lectura de 4 de los 5 sensores. Siendo el sensor de humedad el sensor que no devolvía lecturas en tiempo real de la humedad obtenida en el lugar de la prueba.

Figura 17

Visualización de la información obtenida por los sensores del nodo.



Nota. La figura describe la lectura obtenida por el serial monitor del Arduino UNO de los sensores del nodo.

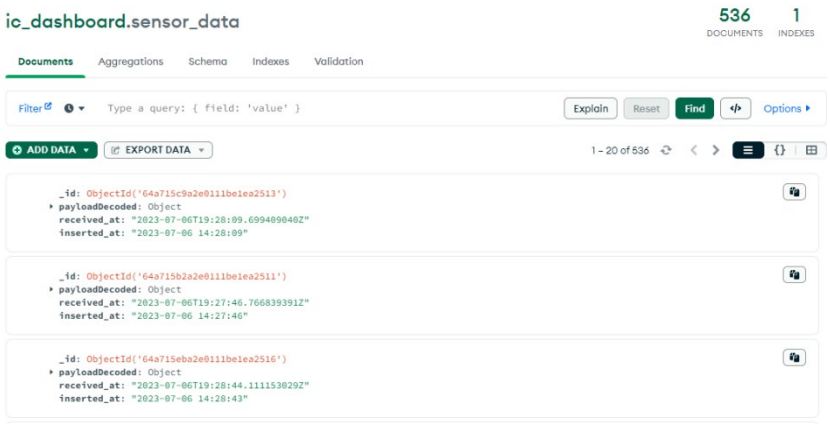
Al analizar este resultado se pudo concluir que el sensor DHT11 utilizado en el nodo implementado es un sensor de una lectura por lo cual este sensor solo registra temperatura y no humedad y temperatura. La siguiente evaluación efectuada consistió en una prueba de conectividad del Gateway del servidor IoT TTN con el nodo de sensores. Esta conexión se realiza a través del módulo Dragino que se encuentra en el nodo de sensores y el Gateway TTN. Se pudo apreciar que la comunicación entre estos dos elementos se realizó de manera exitosa, devolviendo una respuesta encriptada en tiempo real como se puede observar en la Figura 7.

Una vez obtenida la información de Gateway TTN, se realizó una evaluación de la aplicación levantada en la misma nube del Gateway TTN. A través de esta aplicación se conecta el decodificador de señal con el protocolo MQTT y posteriormente se envía a través de un Api REST. Como se puede apreciar en la Figura 9, existe una recepción correcta de la información por parte de la aplicación.

Con la información correctamente obtenida en la aplicación TTN se procedió a evaluar la recepción y decodificación de la información a través de los microservicios. Esto se realizó a través del consumo de una Api REST provista por la aplicación TTN. Para la evaluación de la funcionalidad de los microservicios se utilizó Postman, el cual al momento de llamar al endpoint se obtuvo la información de los sensores correctamente. En los microservicios al momento de realizar la lectura en tiempo real, se realiza también el registro de la información en la base de datos MongoDB ubicado en la nube Atlas, como se puede comprobar en la Figura 16.

Figura 18

Registros de lecturas del nodo obtenidas.

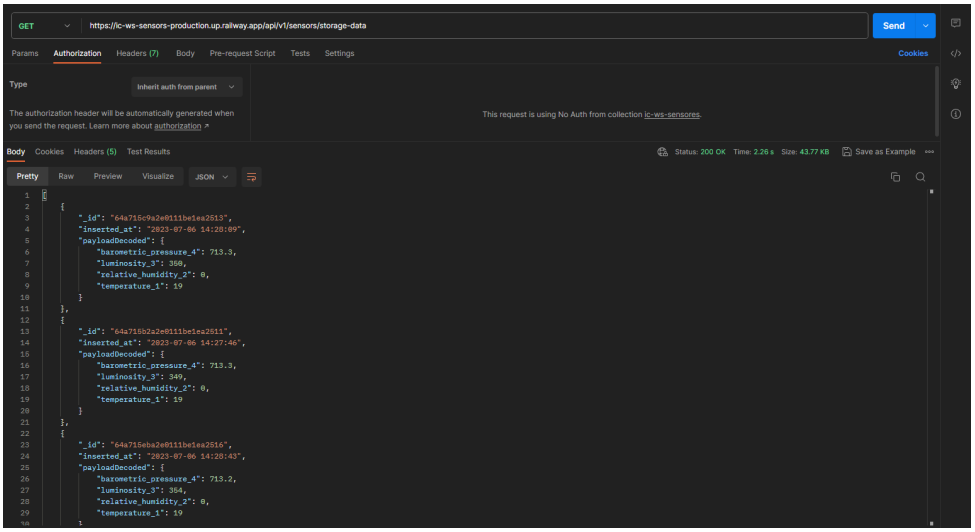


Nota. Registro de las lecturas obtenidas a través del microservicio, almacenadas en la base de datos MongoDB en el servidor en la nube Atlas.

De igual manera se prueban utilizando Postman el consumo de estos registros de la base de datos.

Figura 19

Visualización de la información obtenida al ejecutar el endpoint del microservicio storage-data.

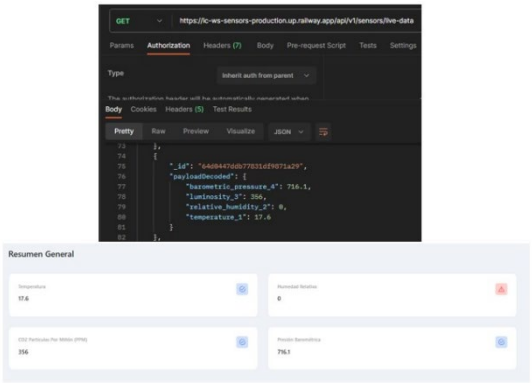


Nota: La figura describe la recepción de información al ejecutar el microservicio de acceso storage-data.

Finalmente se procede a evaluar la visualización de la información en el aplicativo. Para lo cual se despliega el aplicativo y se comprueban los datos mostrados en tiempo real con los datos obtenidos al ejecutar el controlador en Postman simultáneamente.

Figura 20

Prueba de recepción de información en tiempo real del dashboard.



Nota. La figura describe la recepción de información en tiempo real del dashboard comparándolo con la información obtenida desde postman.

Streaming en tiempo real

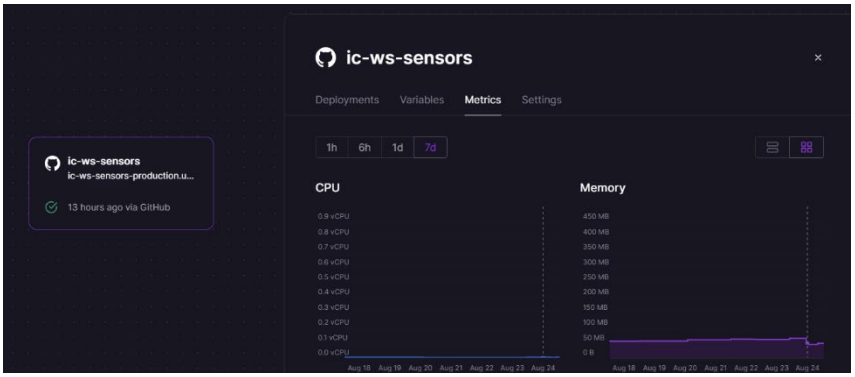
Para el evaluar el streaming en tiempo real se procedió ejecutar la simulación del dron con el Arduino y el ESP32-CAM y se accedió a la dirección IP provista mediante el componente insertado en el front-end. Como se observa en la Fig, se despliega el video streaming en tiempo real, reproduciendo el video obtenido por el nodo simulador como se puede observar en la Figura 15.

Despliegue en la nube del prototipo

Para evaluar el despliegue en la nube, se realizó una implementación haciendo uso del proveedor de IAAS railway. En este proveedor se procedió a levantar los microservicios y el dashboard web. Se realizaron los cambios respectivos en el acceso a los microservicios por parte del front-end y como se puede apreciar en la Figura 19 se realizaron pruebas de despliegue y ejecución.

Figura 21

Despliegue de los microservicios y del dashboard.



Nota. La figura describe la información del despliegue tanto de los microservicios como del dashboard.

Capitulo V: Conclusiones y trabajos futuros

Conclusiones

El presente trabajo de investigación abordó el desafío crítico de mejorar la toma de decisiones en la prevención y gestión de incendios forestales a través del desarrollo de un sistema integral de monitoreo y prevención. A través del cumplimiento de los objetivos planteados, se logró un avance significativo en la creación de un prototipo funcional y eficiente que integra datos de sensores y drons en un dashboard en la nube. A continuación, se presentan las principales conclusiones derivadas de este estudio.

En primer lugar, la investigación sobre metodologías y herramientas de desarrollo de software permitió identificar las mejores prácticas y enfoques más actuales para la construcción del dashboard. Esta exploración abarcó desde la selección de frameworks para el desarrollo frontend hasta la elección de soluciones en la nube para el despliegue del sistema completo. La adopción de enfoques vigentes garantizó la creación de un sistema sólido y escalable, preparado para enfrentar los desafíos cambiantes del entorno tecnológico.

La construcción del prototipo del sistema fue un logro trascendental en este proyecto. La integración exitosa de la lectura de nodos de sensores y la transmisión en tiempo real de la emulación de cámara de un dron dentro de una aplicación web demostró la viabilidad técnica de la propuesta. La interfaz de usuario diseñada en el dashboard proporcionó una experiencia intuitiva y accesible para los usuarios, permitiéndoles visualizar datos cruciales y tomar decisiones informadas de manera eficiente.

La etapa de despliegue e implementación en la nube marcó un paso fundamental hacia la globalización de la información. La capacidad de acceder a la información del sistema desde diferentes plataformas y ubicaciones se convirtió en un activo esencial para la gestión y prevención de incendios forestales. La infraestructura en la nube aseguró la disponibilidad y

escalabilidad del sistema, garantizando que la información vital esté al alcance en cualquier momento y lugar.

La evaluación de la funcionalidad del sistema y la obtención de información, así como el registro histórico, arrojó resultados alentadores. Se comprobó que el prototipo era capaz de capturar y transmitir datos de manera eficiente, brindando información valiosa para la toma de decisiones. La capacidad de registrar históricos permitió analizar patrones y tendencias, contribuyendo a una gestión más efectiva y una prevención más precisa de incendios forestales.

En síntesis, este estudio logró materializar un prototipo funcional y prometedor de un sistema de monitoreo y prevención de incendios forestales a través de un dashboard en la nube. El cumplimiento de los objetivos generales y específicos establecidos sentó las bases para futuros desarrollos en la mejora continua de esta solución. La sinergia entre el uso de tecnologías avanzadas y la aplicación de mejores prácticas en el desarrollo de software ha resultado en un avance significativo en la gestión y mitigación de emergencias ambientales críticas.

Trabajos Futuros

Durante el desarrollo de este trabajo, se han logrado importantes avances en el desarrollo del prototipo de un dashboard integrado en la nube para el sistema de monitoreo y prevención de incendios forestales, con un enfoque en la gestión de información de un nodo específico. Sin embargo, existen diversas oportunidades de mejora y expansión para futuras investigaciones y desarrollos en este campo. Tales como la evaluación y el análisis de la funcionalidad de nuestro prototipo en situaciones reales de incendios forestales, es decir como la implementación del prototipo logró prevenir incendios forestales en áreas que sean vulnerables; y otras que se detallan a continuación.

Una de las principales áreas de mejora para el dashboard es permitir la visualización y gestión de información de toda una red de nodos en lugar de limitarse a un solo nodo. Esta ampliación permitiría una vista más completa y holística del sistema de monitoreo y prevención de incendios forestales, lo que facilitaría la identificación de patrones y tendencias en el comportamiento de la red. Para lograr esto, será necesario implementar técnicas de agregación y visualización de datos que permitan la representación eficiente de grandes volúmenes de información provenientes de múltiples nodos.

Otra mejora clave en la arquitectura es la implementación de un API Gateway, lo que proporcionaría un punto de entrada único para el acceso a los servicios y datos del sistema de monitoreo. El API Gateway simplificaría la comunicación entre los distintos componentes del sistema y permitiría una mejor gestión de la seguridad y la autenticación. Además, facilitaría la integración con otras aplicaciones y sistemas externos, lo que abriría nuevas oportunidades de colaboración y expansión.

Para lograr una mayor escalabilidad, disponibilidad y rendimiento del sistema, se sugiere considerar el despliegue en nubes más profesionales, como Azure u otras plataformas de nube reconocidas. Estas nubes ofrecen servicios avanzados que facilitarían la gestión de recursos, el escalado automático y la alta disponibilidad del sistema. Además, la implementación de prácticas de DevOps, como integración continua (CI/CD), permitiría una entrega de software más ágil y eficiente, lo que agilizaría las actualizaciones y mejoras del sistema.

La dockerización de los componentes del sistema brindaría ventajas en términos de portabilidad y gestión de contenedores. Al encapsular cada componente en contenedores Docker, se garantiza una mayor independencia y aislamiento, lo que facilitaría el despliegue en diferentes entornos y la gestión de versiones. Además, la dockerización puede contribuir a una mayor estabilidad y eficiencia en la ejecución de los servicios.

A futuro, se podría considerar la implementación de técnicas de análisis de big data para procesar y analizar grandes volúmenes de datos generados por el sistema de monitoreo. El uso de tecnologías y herramientas de big data, como Apache Hadoop y Apache Spark, permitiría identificar patrones y tendencias significativas en los datos recopilados, lo que proporcionaría información valiosa para mejorar la eficiencia y la efectividad de las operaciones de prevención de incendios forestales. Además, el análisis de big data podría ayudar a identificar comportamientos anómalos y alertar oportunamente sobre posibles riesgos de incendios.

La incorporación de programación funcional se usaría para agregar nodos al dashboard, esta representaría una mejora significativa para su flexibilidad y escalabilidad. La programación funcional ofrece ventajas en términos de modularidad y reusabilidad del código, lo que facilitaría la adición de nuevos nodos a la red sin afectar la funcionalidad existente. Además, la programación funcional puede mejorar la legibilidad y el mantenimiento del código, lo que es esencial para proyectos en constante evolución.

En resumen, los trabajos futuros de este proyecto abren una perspectiva emocionante para mejorar y expandir el sistema de monitoreo y prevención de incendios forestales. La visualización de la red de nodos, la implementación de un API Gateway, el despliegue en nubes profesionales con DevOps y la dockerización de componentes, el uso de programación funcional para agregar nuevos nodos y la implementación de análisis de big data son áreas de investigación y desarrollo que prometen potenciar la eficiencia, la escalabilidad y la usabilidad del sistema, asegurando así un mayor impacto y utilidad en la prevención y manejo de incendios forestales.

Bibliografía

API Documentation Tool | Postman. (s/f). https://www.postman.com/api-documentation-tool/

Armash Aslam, F., Nabeel Mohammed Jummal Musab Mohd Munir Murade Aaraf Gulamgaus, H., & Lokhande Assistant Professor, P. S. (2015). Efficient Way Of Web Development Using Python And Flask. En *International Journal of Advanced Research in Computer Science* (Vol. 6, Número 2). www.ijarcs.info

Bogotá, M. F. C., Lizcano, V. N., & Quintero, R. A. S. (2023). DISEÑO DE UN PICOSATÉLITE PARA EL MONITOREO Y PREDICCIÓN DEL COMPORTAMIENTO DE INCENDIOS FORESTALES. *Revista Ingeniería, Matemáticas y Ciencias de la Información*. https://api.semanticscholar.org/CorpusID:258230257

Cadavid, A. N., Martínez, J. D. F., & Vélez, J. M. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, *11*(2), 30–39. redalyc.org/pdf/4962/496250736004.pdf

Chinnici, R., Moreau, J.-J., Ryman, A., & Weerawarana, S. (2007). *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. W3C Recommendation. https://www.w3.org/TR/2007/REC-wsdl20-20070626/

De, B. (2017). API Management. *API Management*, 15–28. https://doi.org/10.1007/978-1-4842-1305-6_2

Beynon P. (n.d.). Sistemas de bases de datos. Retrieved August 14, 2023, from https://books.google.com.ec/books?hl=es&lr=&id=XjbeDwAAQBAJ&oi=fnd&pg=PR5&dq=Bases+de+datos&ots=DIEYxIJVLO&sig=u8sW20F7hNWW_bC5f3FPjmOa2nk&redir_esc=y#v=onepage&q=Bases%20de%20datos&f=false

del Busto, Ing. H. G., & Enríquez, Ing. O. Y. (2012). BASES DE DATOS NoSQL. *Telemática*, *11*(3), 21–33. https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/74

Ozierańska, A., Skomra, A., Kuchta, D., & Rola, P. (2016). The critical factors of Scrum implementation in IT project– the case study. Journal of Economics and Management, 25, 79–96. https://doi.org/10.22367/JEM.2016.25.06

Espinosa-Hurtado, R. (2021). Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web. *CEDAMAZ*, *11*(2), 133–141. https://doi.org/10.54753/cedamaz.v11i2.1182

Fowler, M. (2003). Patterns_of_Enterprise_Application_Archi. *Addison-Wesley*, *Ilustrated*, 553.

Grady, B. (2007). Object-oriented Analysis and Design with Applications. *Addison-Wesley*, *3*, 691.

Google. (2023). Angular - Introduction to the Angular docs. Introduction to the Angular Docs. https://angular.io/docs

Guerrero, H., & Carolina, K. (2019). *Sistema monitoreo y alerta temprana de incendios forestales, estructurales y cobertura vegetal en la ciudad de Santa Marta*. https://api.semanticscholar.org/CorpusID:192048191

Ibarra, S. G. P., Quispe, J. R., Mullicundo, F. F., & Lamas, D. A. (2021). Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd. *XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja)*, *August 2021*, 963–968. http://sedici.unlp.edu.ar/handle/10915/120476

Maida, E. G., & Pacienzia, J. (2015). *Metodologías de desarrollo de software*.

MongoDB Documentation. (s/f). https://www.mongodb.com/docs/

Myatt, G. J., & Johnson, W. P. (2009). *Making sense of data II: A practical guide to data visualization, advanced data mining methods, and applications* (Vol. 2). John Wiley \& Sons.

Orellana, C., Patricio, R., Tigse, R., Patricio, F., Parra, M., & Miguel, D. (2021). *Prototipo de sistema de información para alertas tempranas de incendios forestales*.

Pantoja, L., & Pardo, C. (2016). Evaluando la Facilidad de Aprendizaje de Frameworks mvc en el Desarrollo de Aplicaciones Web. *Publicaciones e Investigación*, *10*, 129–142. https://doi.org/10.22490/25394088.1592

Pressman, R. S. (2010). *Ingeniería del software : un enfoque práctico*. McGraw-Hill.

Qian, L., Luo, Z., Du, Y., & Guo, L. (2009). Cloud computing: An overview. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5931 LNCS*, 626–631. https://doi.org/10.1007/978-3-642-10665-1_63/COVER

Railway Docs. (s/f). https://docs.railway.app/

Suwaid, M. M., Habaebi, M. H., & Khan, S. (2019). Embedded LoRaWAN for Agricultural Sensing Applications. *ICETAS 2019 - 2019 6th IEEE International Conference on Engineering, Technologies and Applied Sciences*. https://doi.org/10.1109/ICETAS48360.2019.9117346

Talwana, J. C., & Hua, H. J. (2017). Smart World of Internet of Things (IoT) and Its Security Concerns. *Proceedings - 2016 IEEE International Conference on Internet of Things; IEEE Green Computing and Communications; IEEE Cyber, Physical, and Social Computing; IEEE Smart Data, iThings-GreenCom-CPSCom-Smart Data 2016*, 240–245. https://doi.org/10.1109/ITHINGS-GREENCOM-CPSCOM-SMARTDATA.2016.64

Tatiana Gómez Suárez, K., Anaya, R., & Cano, A. F. (2018). *Un acercamiento a los microservicios*. https://martinfowler.com/mi

VISUALIZACIÓN DE DATOS - Campus2B Bilbao. (s/f). Recuperado el 15 de agosto de 2023, de https://www.campus2b.com/evento/visualizacion-de-datos/

Patricia, J., Gamboa, Z., Alexandra, C., & Arreaga, L. (2018). Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. Evolution of the Methodologies and Models used in Software Development. *INNOVA Research Journal*, *3*(10), 20–33.