Sistema de Monitoreo IoT mediante Python para la Optimización de Cultivos inteligentes

Carold Johanna González Olaya

Universidad Nacional Abierta y a Distancia - UNAD
Escuela de Ciencias Básicas, Tecnologías e Ingenierías
Programa de Ingeniería de Sistemas
CEAD Ibagué - Tolima
2025

Sistema de Monitoreo IoT mediante Python para la Optimización de Cultivos Inteligentes

Carold Johanna González Olaya

Trabajo De Grado Para Optar Por El Título De Ingeniera En Sistemas

Director de Trabajo de Grado
Juan Esteban Tapias

Universidad Nacional Abierta y a Distancia UNAD
Escuela de Ciencias Básicas Tecnologías e Ingenierías
Ibagué - Tolima
2025

Nota de Aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Ibagué, septiembre 2025

Dedicatoria

A mis padres, hermanos y amigos, por su apoyo incondicional, cariño y constante compañía durante todo este proceso, que me permitió alcanzar esta meta con éxito.

Agradecimientos

En primer lugar, agradezco a Dios por Su infinita sabiduría y gracia, que me permitieron culminar con éxito esta etapa. A mi familia y amigos, por su constante apoyo, comprensión y presencia en cada momento de este proceso. Asimismo, al Ingeniero Juan Esteban Tapias, por su valiosa orientación y acompañamiento a lo largo de mi formación académica.

Resumen

Actualmente, estamos inmersos en un entorno donde la tecnología es una parte integral de la sociedad. La llegada de Internet ha impulsado el desarrollo de diversos campos industriales, incluyendo el sector agroindustrial. Según cifras de IDC, este sector ha invertido aproximadamente un 60% en nuevas tecnologías IoT, como sensores de humedad, pH y temperatura. El uso de estos dispositivos ha demostrado mejoras significativas en la producción, mantenimiento y supervisión de los cultivos.

Estas tecnologías han reducido considerablemente los costos asociados con la supervisión y recolección de cultivos. Además, el desarrollo de aplicaciones de software, drones, supervisión remota y minería de datos en el IoT agroindustrial ha resultado en productos de mayor calidad y cultivos más productivos por área.

El objetivo de este proyecto es el montaje de un sistema de monitoreo basado en IoT que recopile datos de sensores distribuidos y los visualice a través de una aplicación web desarrollada en Python.

Palabras clave: iot, cultivos inteligentes, cultivos automatizados, python, api

Abstract

We are currently immersed in an environment where technology is an integral part of society. The advent of the Internet has driven the development of various industrial fields, including the agro-industrial sector. According to IDC figures, this sector has invested approximately 60% in new IoT technologies, such as humidity, pH, and temperature sensors. The use of these devices has shown significant improvements in crop production, maintenance, and monitoring.

These technologies have significantly reduced the costs associated with crop monitoring and harvesting. In addition, the development of software applications, drones, remote monitoring, and data mining in agro-industrial IoT has resulted in higher quality products and more productive crops per area.

The objective of this project is to set up an IoT-based monitoring system that collects data from distributed sensors and displays it through a web application developed in Python.

Keywords: iot, smart crops, automated crops, python, api

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Introducción

En los últimos años, la tecnología ha transformado la agricultura tradicional, dando paso a los cultivos inteligentes, un modelo basado en la integración de sensores, automatización y análisis de datos para optimizar la producción. Entre las tecnologías clave en este avance, el Internet de las Cosas (IoT) permite la recopilación y transmisión de datos en tiempo real, mientras que Python, un lenguaje de programación versátil y accesible, facilita el procesamiento y análisis de esta información, permitiendo una gestión más eficiente de los cultivos.

El uso de Python en cultivos inteligentes es fundamental para el desarrollo de aplicaciones que procesan grandes volúmenes de datos obtenidos de sensores ambientales, sistemas de riego automatizados y drones agrícolas. Gracias a sus bibliotecas especializadas, Python permite el análisis predictivo, la detección temprana de plagas, la optimización del consumo de agua y fertilizantes, y la toma de decisiones basada en datos. Su integración con IoT no solo mejora la eficiencia de la producción agrícola, sino que también contribuye a la sostenibilidad y reducción del impacto ambiental.

Sin embargo, a pesar de los beneficios de esta tecnología, su adopción aún enfrenta desafíos. La falta de conocimiento técnico por parte de los agricultores, la necesidad de infraestructura adecuada y el costo de implementación son barreras que dificultan su expansión. Además, es fundamental evaluar el impacto real de Python en términos de productividad y rentabilidad en los cultivos inteligentes.

Objetivos

Objetivo General

Desarrollar un sistema de monitoreo web en Python para cultivos inteligentes con Internet de las Cosas, utilizando sensores para recopilar datos de variables fundamentales como temperatura y humedad.

Objetivos Específicos

Recolectar información sobre proyectos e investigaciones de tecnologías IoT en sistemas de cultivos inteligentes que hagan uso de Python para su monitoreo y control.

Seleccionar las mejores librerías y herramientas para el almacenamiento de la información suministrada por los sensores que permita el monitoreo y control con Python.

Diseñar un sistema de monitoreo que permita la visualización y análisis de la información.

Planteamiento Del Problema

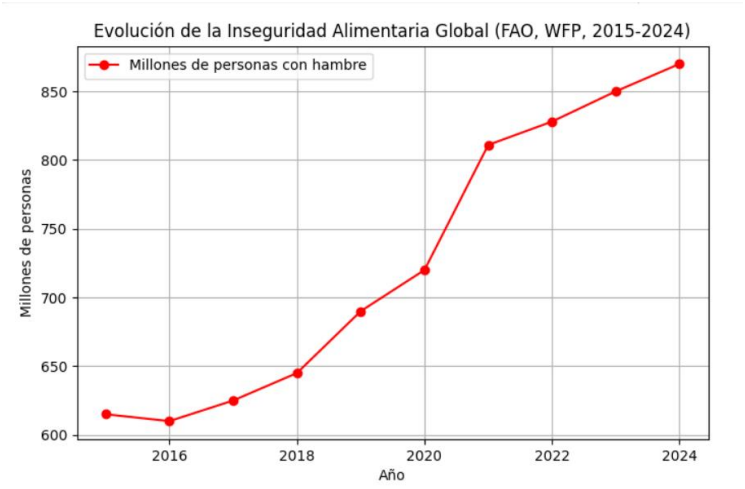
Definición Del Problema

La hambruna global es un fenómeno complejo que impacta a millones de personas en todo el mundo y cuya gravedad ha aumentado significativamente en los últimos años. Según el Programa Mundial de Alimentos (WFP, 2025), más de 343 millones de personas en 74 países enfrentan altos niveles de inseguridad alimentaria. La Organización Panamericana de la Salud (OPS, 2023) también alerta que una de cada 11 personas sufrió hambre, lo que representa un incremento alarmante en comparación con años anteriores.

Como se observa en la Figura 1, la evolución de la inseguridad alimentaria global entre 2015 y 2024 muestra un incremento sostenido en el número de personas afectadas, lo cual evidencia la creciente magnitud de este problema.

Figura 1.

Evolución de la inseguridad Alimentaria Global (FAO,WFP, 2015-2024)



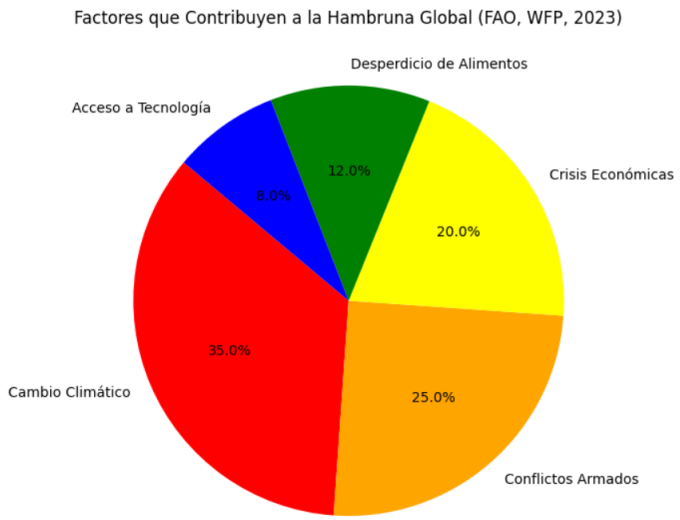
Las causas de esta crisis son diversas y multifactoriales. Entre los principales factores que agravan la producción y distribución de alimentos se encuentran los conflictos armados, desplazamientos forzados, pobreza extrema, choques económicos y el cambio climático.

Como se muestra en la Figura 2, los principales factores que contribuyen al hambre global son el cambio climático (35%), los conflictos armados (25%) y las crisis económicas (20%), seguidos por el desperdicio de alimentos y el acceso limitado a la tecnología.

La directora ejecutiva del WFP, Cindy McCain, destaca que “las necesidades humanitarias globales están aumentando, impulsadas por devastadores conflictos, desastres climáticos más frecuentes y una agitación económica extendida”. Además, según el informe El Estado de la Seguridad Alimentaria y la Nutrición en el Mundo 2021 (FAO, FIDA, OMS, PMA y UNICEF, 2021), en 2020 más de 811 millones de personas padecieron hambre crónica, y aproximadamente 900,000 estuvieron en condiciones cercanas a la hambruna.

Figura 2.

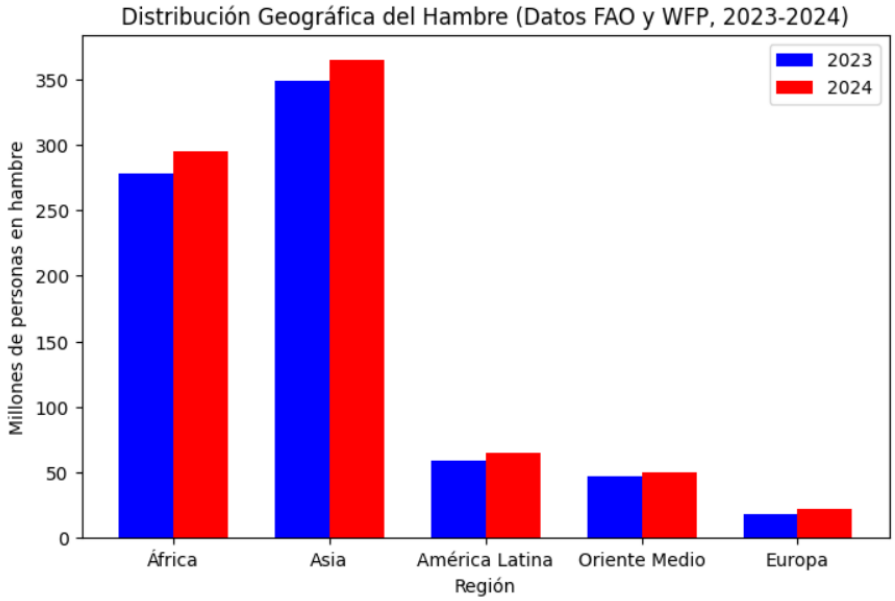
Factores que Contribuyen a la Hambruna Global.



De igual manera, la Figura 3 evidencia la distribución geográfica del hambre, donde Asia y África concentran la mayor proporción de personas afectadas, mientras que América Latina, Oriente Medio y Europa presentan cifras relativamente menores, pero igualmente significativas.

Figura 3.

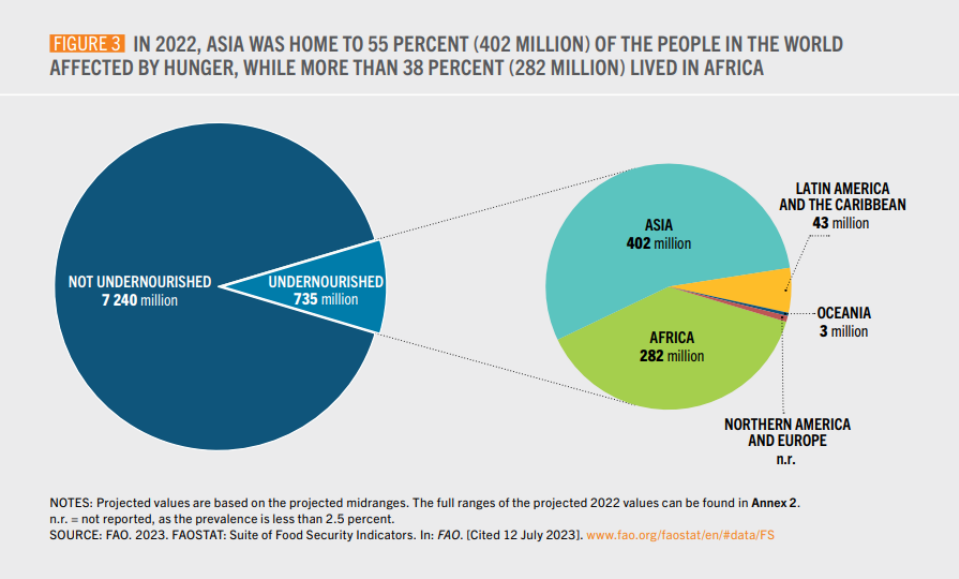
Distribución Geográficas del Hambre (Datos FAO y WFP, 2023 – 2024)



El año 2025 se proyecta como un periodo crítico en la crisis alimentaria global. Este problema no solo afecta la salud y el desarrollo de las poblaciones vulnerables, sino que también impacta negativamente la economía global, al reducir la productividad laboral y limitar el crecimiento económico de los países afectados (Hernández, Correa & Correa, 2018). La inseguridad alimentaria genera una espiral de pobreza y desigualdad, afectando particularmente a países con economías dependientes de la agricultura. Tal como se muestra en la Figura 4, el porcentaje de personas en el mundo afectadas por el hambre continúa en aumento, consolidando una tendencia alarmante.

Figura 4.

Porcentaje de personas en el mundo afectadas por el hambre



El caso de Colombia y la necesidad de soluciones tecnológicas

En Colombia, la agricultura ha sido históricamente un sector fundamental para la economía y la seguridad alimentaria. En 2020, este sector representó el 6.8% del PIB, y para 2024, su impacto creció hasta el 9% en el segundo trimestre del año (UPRA, 2024). Cultivos estratégicos como el café, el plátano y el maíz son esenciales para el abastecimiento interno y la exportación. Sin embargo, los efectos del cambio climático, la degradación del suelo, el acceso limitado a tecnología agrícola y la desigualdad en la distribución de recursos han afectado gravemente la producción (Tapias, 2021).

El crecimiento poblacional en Colombia ha incrementado la demanda de alimentos. Según Tapias (2021), entre 2005 y 2017 la población aumentó en un 13.8%, pasando de 41.6 a 47.4 millones de habitantes, con proyecciones que indican un crecimiento del 23.5% para 2070. A pesar de este crecimiento, solo 7.1 millones de hectáreas están cultivadas, frente a los 22

millones de hectáreas disponibles para la agricultura. Además, el 80% de la tierra cultivable está destinada a la ganadería, reduciendo aún más la producción de alimentos esenciales.

Otro desafío clave es la infraestructura agrícola deficiente. Según Tapias (2021), en 2015 solo el 7% de las vías rurales donde se genera la producción agrícola estaban pavimentadas, dificultando el transporte y comercialización de productos. Además, la mano de obra en el sector agrícola se ha reducido debido al envejecimiento del campesinado y la falta de interés de las nuevas generaciones en la actividad agrícola.

El Plan Nacional de Seguridad Alimentaria y Nutricional (PNSAN) ha implementado estrategias para promover la agricultura sostenible y el acceso equitativo a alimentos saludables (DANE, 2020). Sin embargo, los desafíos actuales requieren medidas más innovadoras, como inversiones en tecnología agrícola avanzada, desarrollo de infraestructura y la aplicación de herramientas de automatización y análisis de datos.

El papel de la tecnología en la seguridad alimentaria

Ante esta crisis, la integración de tecnologías emergentes como el Internet de las Cosas (IoT), la inteligencia artificial y el análisis de datos con Python se plantea como una solución viable para mejorar la productividad agrícola y mitigar la inseguridad alimentaria. Estas tecnologías pueden contribuir a optimizar procesos en el sector agrícola mediante:

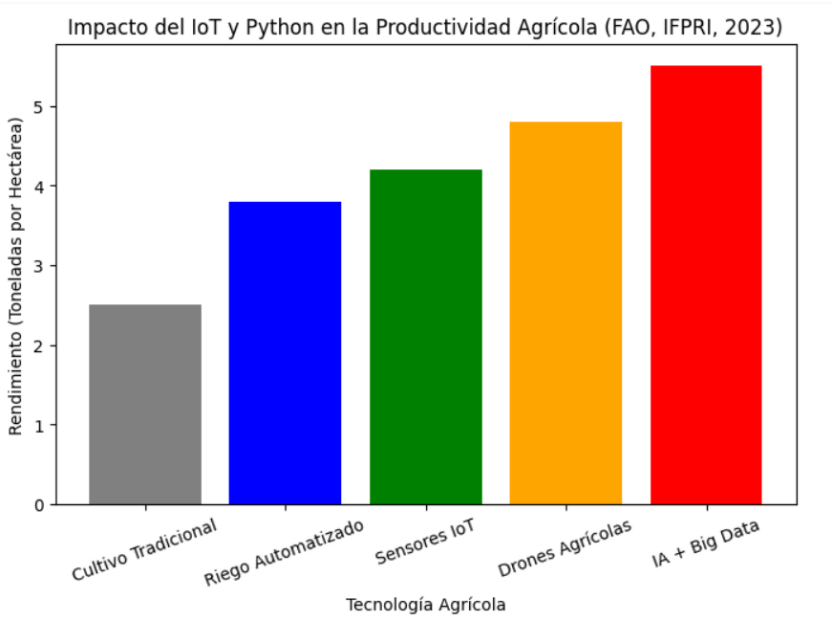
- Sensores IoT para monitorear cultivos en tiempo real y gestionar de manera eficiente el uso del agua y fertilizantes.
- Sistemas de automatización agrícola para mejorar la rentabilidad de los cultivos con menor impacto ambiental.

- Plataformas de análisis de datos con Python, que permiten recopilar y procesar información clave para la toma de decisiones basadas en predicciones meteorológicas, rendimiento de cultivos y estrategias de cultivo de precisión.

Como se aprecia en la Figura 5, el impacto del IoT y Python en la productividad agrícola representa una oportunidad significativa para enfrentar la inseguridad alimentaria global, al incrementar la eficiencia y la sostenibilidad de los cultivos.

Figura 5.

Impacto de IoT y Python en la Productividad Agrícola (FAO, IFPRI, 2023)



Un ejemplo del impacto positivo de la tecnología en la agricultura es el uso de cultivos inteligentes, que dependen de sistemas avanzados de monitoreo y control. Los cultivos aeropónicos son ejemplo de cultivos inteligentes o automatizados y según Tapias (2021), estos métodos han demostrado ventajas como el ahorro de agua, mayor velocidad de producción, reducción de plagas y la posibilidad de cultivos en entornos urbanos sin necesidad de tierra. Casos internacionales, como la granja Mirai en Japón, donde se producen más de 10,000

lechugas al día con IoT, y Aerofarms en EE.UU., demuestran el potencial de estas tecnologías en la agricultura moderna.

A pesar de los beneficios potenciales de estas soluciones, su adopción enfrenta desafíos importantes. Los altos costos de implementación, la falta de infraestructura adecuada y la carencia de conocimientos técnicos en el sector agrícola dificultan su aplicación en diversas escalas de producción. Por lo tanto, es esencial explorar estrategias que permitan la democratización y accesibilidad de estas tecnologías, asegurando su implementación efectiva en diferentes contextos agrícolas.

Por lo anterior, la pregunta clave es: ***"¿Cómo podemos desarrollar soluciones de monitoreo agrícola que sean sostenibles y utilicen tecnologías emergentes para abordar la inseguridad alimentaria?"***

Justificación

El avance tecnológico ha revolucionado múltiples sectores, y la agricultura no es la excepción. En este contexto, el Internet de las Cosas (IoT) ha surgido como una herramienta clave en la modernización del sector agrícola, permitiendo la interconexión de dispositivos y la recopilación de datos en tiempo real a través de sensores inteligentes (Chanchí, Ospina, & Saba, 2022). Una de las aplicaciones más innovadoras de esta tecnología es en los sistemas de cultivo aeropónico, donde las plantas crecen con sus raíces suspendidas en el aire y reciben nutrientes mediante aspersión controlada (Carvajal, J. 2019).

Los cultivos inteligentes y automatizados, apoyados en tecnologías como sensores IoT, análisis de datos y sistemas de control automatizado, ofrecen ventajas significativas sobre los métodos tradicionales de agricultura. Estos sistemas permiten monitorear y regular de forma precisa variables críticas como la temperatura, humedad, pH y niveles de nutrientes, optimizando el uso de recursos como el agua, con ahorros que pueden alcanzar hasta el 90% en comparación con la agricultura convencional (Salazar, 2021). Además, su capacidad para operar en cualquier entorno y durante todo el año, sin depender de factores climáticos o del tipo de suelo, los hace especialmente adecuados para su implementación en zonas urbanas o regiones con condiciones adversas para el cultivo tradicional.

A nivel global, países como Corea, China, Rwanda y Estados Unidos han comenzado a incorporar estas tecnologías en cultivos estratégicos como la papa, el tomate y la lechuga (Otazú, 2010). Empresas como Bowery y AppHarvest están liderando esta transformación, invirtiendo en infraestructuras altamente tecnificadas que permiten una producción más eficiente y sostenible (Severson, 2021). Sin embargo, a pesar de sus múltiples beneficios, la adopción de la agricultura

inteligente sigue siendo limitada debido a barreras como los altos costos iniciales, la falta de conocimiento técnico y la resistencia al cambio dentro del sector agrícola (Tapias, 2021).

En este contexto, el uso de Python como herramienta de programación y análisis de datos en sistemas IoT agrícolas se convierte en una solución viable para optimizar los procesos de monitoreo y control de cultivos inteligentes. Python permite procesar grandes volúmenes de datos en tiempo real, facilitando la toma de decisiones mediante algoritmos de aprendizaje automático, visión por computadora y análisis predictivo. Librerías como Pandas, NumPy, TensorFlow y OpenCV permiten la creación de sistemas avanzados que mejoran la productividad agrícola y la eficiencia en la gestión de recursos (Tapias, 2021).

La implementación de IoT en la agricultura moderna se basa en el uso de sensores inalámbricos que recopilan datos en tiempo real sobre humedad, temperatura, pH y otros factores críticos para la salud de los cultivos. Estos datos son enviados a servidores a través de tecnologías como LoRaWAN, que permite la comunicación a larga distancia con bajo consumo de energía (Carvalho, Rodrigues, Alberti, & Solic, 2017). Para la visualización y análisis de la información, se utilizan sistemas de software y API, los cuales facilitan la integración de dispositivos y la automatización de procesos (Microsoft, 2023; Márquez, Araujo, & Royero, 2021; RedHat, 2022).

La FAO resalta que la mecanización y automatización de la agricultura pueden mejorar la productividad, reducir el trabajo manual pesado y contribuir a la sostenibilidad del sector (FAO, 2023). En Colombia, un país donde la agricultura representa un pilar fundamental de la economía, la adopción de tecnologías como IoT y análisis de datos con Python es crucial para mejorar la eficiencia del sector. Aunque tradicionalmente el país ha dependido de cultivos como

el café, caña de azúcar y arroz, en los últimos años ha comenzado una transformación digital en la agricultura, adoptando innovaciones como el uso de drones, sensores de riego inteligente y plataformas de Big Data para la toma de decisiones (SEIDOR, 2023; AgroTech Campus, s.f.).

La automatización en cultivos aeropónicos e hidropónicos se ha beneficiado del uso de IoT y Python. Por ejemplo, en un estudio sobre la automatización de cultivos aeropónicos de cilantro, se implementó un sistema autónomo utilizando Matlab-Simulink-MPLAB para el control del riego y el suministro energético. Los resultados mostraron que la automatización permitió mantener plantas uniformes durante el periodo de pruebas, evidenciando mejoras en el desarrollo de los cultivos aeropónicos (Tapias, 2021). En Colombia, el programa Agro 4.0, desarrollado por el Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC) y el Centro para la Cuarta Revolución Industrial (C4IR.CO), busca mejorar la productividad del sector agropecuario mediante la implementación de tecnologías avanzadas como IoT, inteligencia artificial y computación en la nube. Este programa ha implementado pilotos en diez municipios, beneficiando a cien agricultores en cultivos de café, cacao y aguacate (MinTIC, 2023).

La implementación de IoT y Python en la agricultura ofrece múltiples beneficios:

- **Eficiencia en el Uso de Recursos:** La automatización permite un control preciso del riego y la nutrición, optimizando el uso de agua y nutrientes.
- **Mejora en la Productividad:** Al mantener condiciones óptimas de crecimiento, se incrementa la producción y calidad de los cultivos.

- **Reducción de Pérdidas:** La monitorización constante ayuda a detectar y corregir problemas de manera oportuna, disminuyendo pérdidas por enfermedades o condiciones adversas.

Sin embargo, a pesar de sus beneficios, la adopción de estas tecnologías enfrenta desafíos:

- **Costos Iniciales:** La inversión en tecnología puede ser elevada para pequeños agricultores.
- **Capacitación:** Es necesario capacitar a los agricultores en el uso y mantenimiento de estas tecnologías.
- **Infraestructura:** La falta de infraestructura adecuada puede limitar la implementación efectiva de sistemas IoT.

Colombia ha implementado proyectos innovadores que integran IoT, análisis de datos y automatización en el sector agrícola, mejorando la productividad y sostenibilidad del campo. Algunos ejemplos incluyen:

- Drones y sensores para la agricultura de precisión, utilizados en cultivos de café y caña de azúcar para monitorear la salud de las plantas y optimizar el uso de insumos (Ministerio de Agricultura y Desarrollo Rural de Colombia, 2020; Fedecafé, 2021).
- Sistemas de riego inteligente en la región Caribe, que usan sensores de humedad y riego automatizado para maximizar la eficiencia del agua en cultivos como el arroz y hortalizas (Agrosavia, 2019).

- Big Data en la agricultura, aplicado en cultivos de palma de aceite para la toma de decisiones basada en el análisis de datos meteorológicos, plagas y ciclos de producción (Agrosavia, 2018).
- Blockchain en la comercialización de café, garantizando la trazabilidad del producto y mejorando la transparencia en las cadenas de suministro (Bolaños & Rodríguez, 2021).

Estos avances han demostrado que la combinación de IoT y Python en la agricultura permite una gestión eficiente de cultivos, reducción de costos y mayor sostenibilidad.

Marco Teórico

Definición y características

Los cultivos inteligentes representan una evolución de la agricultura tradicional mediante la integración de tecnologías avanzadas como el Internet de las Cosas (IoT), Big Data, inteligencia artificial (IA), sensores y automatización. Estas tecnologías permiten optimizar la producción de alimentos de manera eficiente, sostenible y con menor impacto ambiental (FAO, 2023). Su objetivo es maximizar los rendimientos agrícolas, reducir el consumo de recursos y mejorar la adaptabilidad de los cultivos a condiciones climáticas cambiantes.

El concepto de cultivos inteligentes surge como respuesta a la necesidad de incrementar la producción de alimentos ante el crecimiento poblacional y el cambio climático. Históricamente, la revolución verde de mediados del siglo XX introdujo mecanización y agroquímicos para mejorar la productividad. Sin embargo, el actual enfoque de la "agricultura 4.0" busca ir más allá, incorporando tecnologías digitales para un control preciso de los cultivos (FAO, 2021).

Características de los cultivos inteligentes:

Los cultivos inteligentes se consolidan como un enfoque innovador dentro de la agricultura de precisión, al integrar tecnologías digitales, automatización y análisis de datos en los procesos productivos. Su propósito es optimizar el uso de recursos.

- **Automatización y monitoreo en tiempo real:** Mediante sensores y dispositivos IoT, se recopilan datos en tiempo real sobre temperatura, humedad, niveles de nutrientes y condiciones del suelo. Estos datos se utilizan para realizar ajustes automáticos en los sistemas de riego y fertilización, reduciendo desperdicios y optimizando el crecimiento de los cultivos (Carvalho et al., 2017). Por ejemplo, en los invernaderos de Almería, España, se utilizan sistemas

de control automatizado para ajustar la temperatura y humedad, mejorando la eficiencia del riego hasta en un 40% (González, 2022).

- **Optimización del uso de recursos:** A través de la implementación de sistemas de riego inteligente y fertilización controlada, se logra una reducción significativa en el consumo de agua y fertilizantes. Por ejemplo, técnicas como la aeroponía y la hidroponía pueden reducir el uso de agua en un 90% en comparación con la agricultura convencional (Salazar, 2021). Un caso exitoso es el sistema de riego automatizado en California, donde los agricultores han reducido el consumo de agua en cultivos de fresa en un 35% mediante sensores IoT (Smith et al., 2020).
- **Toma de decisiones basada en datos:** La integración de Big Data e inteligencia artificial permite la recolección, a macenamiento y análisis de grandes volúmenes de datos, optimizando la planificación de cultivos y la detección temprana de plagas y enfermedades (Microsoft, 2023). En Brasil, el uso de plataformas de análisis predictivo ha incrementado en un 20% la productividad en cultivos de soja al detectar patrones climáticos y ajustar estrategias de siembra (AgroTech Campus, 2022).
- **Adaptabilidad y resiliencia:** Los cultivos inteligentes pueden operar en diversos entornos, incluidos invernaderos, zonas áridas y espacios urbanos, aumentando la producción en lugares con recursos limitados. Por ejemplo, en los Emiratos Árabes Unidos, donde las condiciones climáticas son extremas, se han implementado cultivos en contenedores modulares con iluminación LED y control de temperatura, permitiendo el crecimiento de vegetales en condiciones inhóspitas (FAO, 2021).
- **Mejora en la productividad y calidad**: Gracias a la monitorización constante y la automatización de procesos, se garantiza un mejor control de calidad y mayores rendimientos en la producción agrícola. Un ejemplo de esto es el uso de drones en la producción de café en

Colombia, donde las imágenes aéreas ayudan a identificar áreas con deficiencia de nutrientes, permitiendo aplicar fertilizantes de manera localizada y reduciendo costos en un 15% (Ministerio de Agricultura y Desarrollo Rural de Colombia, 2020).

Estas características hacen que los cultivos inteligentes sean una solución clave para enfrentar los retos del futuro en la producción de alimentos, combinando tecnología y sostenibilidad para maximizar la eficiencia agrícola.

Tecnologías utilizadas en la agricultura inteligente

La agricultura inteligente se apoya en una serie de tecnologías que permiten la automatización, el monitoreo y la optimización de los cultivos. A continuación, se detallan las principales herramientas utilizadas en este ámbito.

a) Sensores IoT (Internet de las Cosas)

Los sensores IoT juegan un papel fundamental en la recopilación de datos en tiempo real sobre el estado de los cultivos. Estos dispositivos permiten detectar factores críticos como la humedad del suelo, temperatura, pH, niveles de CO_2, intensidad lumínica, entre otros (Chanchí, Ospina & Saba, 2022).

- Sensores de humedad del suelo: Utilizados en sistemas de riego inteligente para medir la cantidad de agua disponible en el suelo y evitar el desperdicio (Otazú, 2010).
- Sensores de temperatura y humedad ambiental: Monitorizan el clima dentro de los invernaderos y ajustan las condiciones automáticamente (Carvajal, 2019).
- Sensores de nutrientes: Analizan la composición del suelo y determinan si se requieren fertilizantes adicionales (FAO, 2023).

- Sensores de pH: Permiten monitorear la acidez del suelo para garantizar el crecimiento óptimo de los cultivos (Marquez et al., 2021).
- Sensores de CO_2 y luminosidad: Ayudan a regular la fotosíntesis y optimizar la producción agrícola en ambientes controlados.

Estos sensores están conectados a plataformas de análisis que procesan la información y permiten actuar de manera automatizada y predictiva, optimizando el rendimiento de los cultivos (Carvalho et al., 2017).

Tipos de plataformas de análisis de datos para sensores IoT

Los sensores IoT en la agricultura inteligente están conectados a diversas plataformas que procesan la información y permiten la toma de decisiones automatizadas. Entre las más utilizadas se encuentran:

- **Plataformas en la nube (Cloud Computing):** Servicios como AWS IoT, Microsoft Azure IoT y Google Cloud IoT permiten el almacenamiento, análisis y visualización de datos en tiempo real, facilitando el acceso remoto y la integración con algoritmos de inteligencia artificial (Botta et al., 2016). Estas plataformas disponen de SDKs y bibliotecas para Python —como *boto3* (AWS), azure-iot-device (Azure) y google-cloud-iot (Google Cloud)— que permiten la integración de sensores, la gestión de dispositivos y la implementación de modelos de análisis predictivo en la nube (Botta et al., 2016). Asimismo, existen soluciones más ligeras y orientadas a la investigación y prototipado, como ThingSpeak, una plataforma desarrollada por MathWorks que permite recolectar datos IoT en tiempo real, analizarlos con MATLAB y visualizarlos mediante paneles interactivos. ThingSpeak puede integrarse fácilmente con Python a través de la

librería *requests* o mediante APIs REST, ofreciendo una alternativa práctica y de bajo costo para proyectos de monitoreo de cultivos inteligentes (MathWorks, 2023).

- **Plataformas de análisis de datos agrícolas:** Herramientas como FarmBeats de Microsoft y Climate FieldView de Bayer ofrecen soluciones específicas para la agricultura, permitiendo monitorear cultivos, predecir rendimientos y optimizar el uso de recursos (Wolfert et al., 2017). A través de APIs, los datos generados por estas plataformas pueden ser procesados en Python, facilitando tareas como la predicción de rendimientos, el análisis de tendencias climáticas y la optimización del uso de recursos mediante librerías como pandas, scikit-learn y statsmodels (Wolfert et al., 2017).
- **Plataformas de código abierto:** Sistemas como ThingsBoard, OpenAg y Node-RED promueven el desarrollo de soluciones personalizadas y de bajo costo para el monitoreo agrícola también permiten la personalización de soluciones de monitoreo y automatización agrícola, integrando sensores IoT con herramientas de análisis de datos (Zhang et al., 2020). Python puede emplearse en el backend de ThingsBoard para crear reglas de procesamiento, mientras que en proyectos como OpenAg se utilizó como lenguaje principal para controlar sensores y actuadores. Por su parte, Node-RED, aunque basado en JavaScript, admite la integración de scripts Python mediante nodos especializados, lo que permite combinar flujos visuales con análisis computacional avanzado (Zhang et al., 2020).
- **Plataformas basadas en Edge Computing:** Tecnologías como AWS Greengrass y EdgeX Foundry permiten procesar datos localmente en el campo, reduciendo la latencia y dependencia de la conectividad a Internet. En estos entornos, Python puede utilizarse para ejecutar funciones Lambda o desarrollar microservicios que

realicen procesamiento de datos, generación de alertas o control de actuadores directamente en el dispositivo (Shi et al., 2016).

- **Sistemas de gestión agrícola (FMS - Farm Management Systems):** Software como AgLeader, Granular y AgriWebb centralizan los datos de sensores, maquinaria y condiciones climáticas para optimizar la planificación y operación agrícola (Jayashree et al., 2018). Aunque muchas de estas soluciones son comerciales y de código cerrado, algunas ofrecen APIs que permiten la integración con aplicaciones desarrolladas en Python. Esto posibilita la automatización de procesos de recolección de datos, generación de reportes y análisis estadístico para apoyar la toma de decisiones agronómicas (Jayashree et al., 2018).

b) Redes de comunicación y transmisión de datos

Para garantizar la conectividad y el intercambio de información entre sensores, servidores y aplicaciones, se utilizan diversas tecnologías de comunicación:

- LoRaWAN (Long Range Wide Area Network): Permite transmitir datos a grandes distancias con bajo consumo de energía, ideal para grandes plantaciones (Microsoft, 2023). Python se integra con LoRaWAN a través de bibliotecas como **PyLoRa** y plataformas como **The Things Network**, facilitando el envío de datos de sensores a la nube para su análisis en tiempo real.

  Ejemplo: En Brasil, se ha implementado una red de **sensores LoRaWAN en cultivos de café**, controlados con Python, para monitorear la humedad y prevenir enfermedades en las plantas (FAO, 2023).

- Wi-Fi y Bluetooth: Son más adecuados para cultivos en espacios controlados, como invernaderos o laboratorios agrícolas (RedHat, 2022).

- Redes celulares (4G/5G): Se emplean en zonas con acceso a redes móviles para garantizar la conectividad en tiempo real (Carvajal, 2019).
- Redes satelitales: Utilizadas en zonas rurales o remotas donde no hay acceso a redes terrestres, facilitando la recolección de datos en tiempo real.

c) Sistemas de análisis de datos y machine learning con Python

Python se ha convertido en una herramienta clave en la agricultura inteligente debido a su capacidad para procesar grandes volúmenes de datos, automatizar procesos y desarrollar modelos de predicción basados en inteligencia artificial. Su versatilidad y amplia gama de bibliotecas especializadas lo hacen ideal para diversas aplicaciones agrícolas.

- **Big Data en la agricultura**: Python permite gestionar grandes volúmenes de datos recolectados por sensores IoT, generando modelos predictivos que optimizan el rendimiento de los cultivos. Librerías como Pandas y Dask facilitan el procesamiento y análisis de estos datos (Microsoft, 2023).
- **Automatización del monitoreo de cultivos:** Gracias a la integración con sensores, Python permite la recopilación y análisis de datos en tiempo real, facilitando la toma de decisiones automatizadas para el riego, fertilización y detección de plagas.
- **Algoritmos de inteligencia artificial:** Se utilizan modelos de machine learning para analizar patrones climáticos, identificar deficiencias en el suelo y optimizar la gestión del agua. Algunas de las librerías más utilizadas incluyen TensorFlow, Scikit-learn y PyTorch.
- **Visión por computadora**: Con OpenCV y TensorFlow, Python permite el análisis de imágenes capturadas por drones y cámaras en cultivos, identificando plagas, enfermedades y condiciones adversas de manera automatizada.

- **Automatización de sistemas de riego con Python:** Python, a través de librerías como PySerial y PyModbus, facilita la integración con controladores de riego para gestionar el uso del agua de manera eficiente.

Ejemplo: En Brasil, se ha desarrollado un sistema basado en machine learning con Python para detectar plagas en cultivos de soja, logrando una reducción del 30% en el uso de pesticidas (FAO, 2023), (Silva et al., 2020).

d) Drones y visión por computadora con Python

Los drones agrícolas han revolucionado el monitoreo de cultivos al permitir:

- Detección de plagas y enfermedades a través de cámaras multiespectrales.
- Análisis del estado del suelo y la vegetación para optimizar la siembra y fertilización.
- Fumigación automatizada, reduciendo costos y mejorando la eficiencia en la aplicación de agroquímicos (Ministerio de Agricultura y Desarrollo Rural de Colombia, 2020).

Python es ampliamente utilizado en la programación de drones y el análisis de imágenes capturadas. Librerías como DroneKit, OpenCV y Scikit-image permiten la automatización de vuelos, la captura de imágenes y la detección de patrones en cultivos.

Ejemplos de proyectos destacados

Empresas como AppHarvest y Bowery en EE.UU. emplean drones con inteligencia artificial y Python para analizar el estado de sus cultivos y mejorar su calidad (Severson, 2021).

1. **Proyecto AgroSmart (Brasil)**
    - Uso de sensores IoT y machine learning con Python para optimizar el uso del agua y reducir costos de producción en cultivos de caña de azúcar.

- Aplicación de modelos de predicción con TensorFlow para anticipar sequías y mejorar la planificación del riego.
- Impacto: Reducción del consumo de agua en un 40% y aumento del rendimiento de los cultivos en un 20%.

2. **FarmBeats de Microsoft (EE.UU.)**
    - Plataforma basada en Azure IoT que utiliza Python para procesar datos de sensores de suelo, drones y estaciones meteorológicas.
    - Implementación de algoritmos de deep learning con PyTorch para analizar la salud de los cultivos mediante imágenes satelitales.
    - Impacto: Mejora en la precisión de predicción de rendimiento de cultivos en un 30%.
3. **AgriSense (India)**
    - Uso de visión por computadora con OpenCV y TensorFlow para detectar enfermedades en cultivos de tomate.
    - Algoritmos de machine learning entrenados con Python para identificar patrones de deficiencia nutricional en el follaje.
    - Impacto: Reducción del 25% en pérdidas por enfermedades y optimización del uso de fertilizantes.
4. **Plataforma PyAgro (España)**
    - Sistema de monitoreo agrícola con Python y sensores IoT basado en Raspberry Pi para pequeños agricultores.
    - Integración con librerías como Pandas y Matplotlib para la visualización de datos sobre humedad del suelo y temperatura.

- Impacto: Aumento del 15% en la productividad mediante la toma de decisiones basadas en datos en tiempo real.

5. **Automated Greenhouses (Países Bajos)**

- Implementación de invernaderos inteligentes con redes neuronales en Python para controlar automáticamente la temperatura, humedad y ventilación.
- Integración con sensores IoT y plataformas en la nube como AWS para procesamiento de datos en tiempo real.
- Impacto: Reducción del 30% en costos de energía y aumento del 20% en la producción de hortalizas.

e) Sistemas de riego inteligente con Python

El riego inteligente permite reducir el consumo de agua mediante el uso de sensores IoT y análisis de datos. Los sistemas más utilizados incluyen:

- **Riego por goteo automatizado**: Libera agua en cantidades exactas, evitando el desperdicio.
- **Riego por aspersión con control de sensores**: Se activa solo cuando el suelo alcanza un nivel crítico de sequedad.
- **Aeroponía e hidroponía**: Métodos que ahorran hasta un 90% de agua, ya que las raíces de las plantas reciben nutrientes a través de la nebulización o disoluciones minerales (Salazar, 2021).

La automatización del riego es una aplicación clave dentro de la agricultura inteligente, permitiendo el uso eficiente del agua, la mejora en la salud de los cultivos y la reducción de costos. Python es una herramienta fundamental en este ámbito debido a su capacidad para

interactuar con sensores, controlar actuadores, analizar datos y facilitar la integración con plataformas en la nube y sistemas industriales.

- **Integración con sensores y actuadores**

Python permite la lectura de sensores de humedad del suelo, temperatura, presión y más, utilizando bibliotecas como:

  - PySerial: Para la comunicación con microcontroladores como Arduino.
  - Adafruit_DHT, smbus: Para leer sensores conectados a una Raspberry Pi.

Con estos datos, es posible activar válvulas solenoides o bombas de riego de forma automatizada.

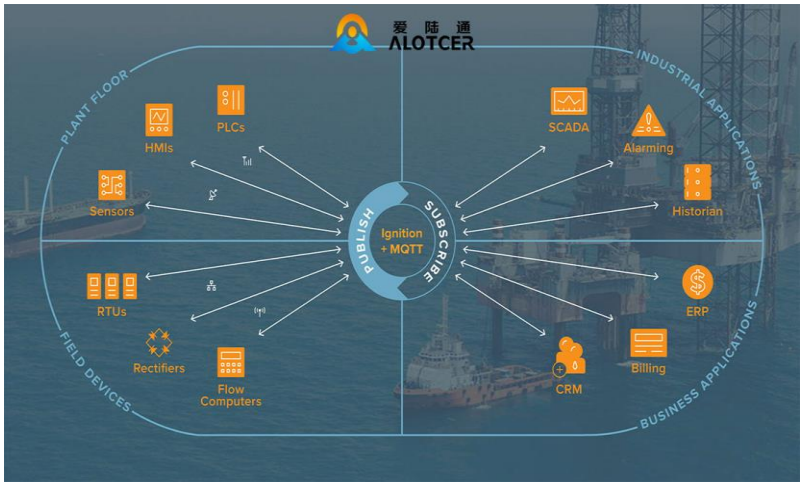
- **Protocolos industriales y comunicación**

Python también facilita la comunicación con dispositivos industriales y sistemas SCADA mediante:

  - PyModbus: Para el protocolo Modbus RTU/TCP utilizado en PLCs.
  - paho-mqtt: Para integrar sensores con plataformas en la nube mediante el protocolo MQTT.

Figura 6.

Protocolos industriales y de Comunicación



Esto permite el control y monitoreo remoto del sistema de riego.

- **Toma de decisiones y automatización**

Mediante bibliotecas como Pandas, Scikit-learn y TensorFlow, es posible analizar datos históricos y climáticos para:

- Predecir la necesidad de riego.
- Calcular el volumen óptimo de agua.
- Ajustar el calendario de riego según pronósticos meteorológicos

- **Interfaces gráficas y visualización**

Python permite crear aplicaciones web y paneles de visualización mediante:

- Flask, FastAPI: Para aplicaciones web ligeras.
- Plotly, Dash, Matplotlib: Para generar gráficos de humedad del suelo, consumo de agua, entre otros.

Ejemplos de otros proyectos destacados:

1. **SmartWater (España)**: Implementación de sensores IoT y algoritmos de machine learning en Python para predecir las necesidades hídricas de los cultivos en tiempo real.
2. **AquaCropNet (EE.UU.)**: Uso de redes neuronales con Python para optimizar el riego en viñedos y reducir el desperdicio de agua.
3. **Sistema de Riego Inteligente en la región Caribe colombiana**: Aplicación de sensores y modelos de predicción con Python en cultivos de arroz y hortalizas, mejorando la eficiencia en el uso del agua (Agrosavia, 2019).
4. **AgriWater (India)**: Plataforma basada en IoT y análisis de datos en la nube para optimizar el riego en plantaciones de arroz.

Importancia de Python en la Automatización del Riego

- Versatilidad: Compatible con múltiples sensores y controladores.
- Escalabilidad: Desde pequeñas parcelas hasta grandes extensiones.
- Open Source: Reducción de costos de implementación.
- Integración con IA: Mejora en la toma de decisiones y eficiencia hídrica.

Compatibilidad de Python con microcontroladores para la agricultura inteligente.

La programación de microcontroladores desempeña un papel fundamental en la automatización y monitoreo de cultivos en tiempo real. Python, gracias a su sintaxis sencilla, comunidad activa y extenso ecosistema de bibliotecas, se ha consolidado como uno de los lenguajes más utilizados en el desarrollo de sistemas embebidos aplicados a la agricultura inteligente. Esta compatibilidad permite una integración fluida entre sensores, actuadores,

servicios en la nube y algoritmos de inteligencia artificial, facilitando la creación de soluciones escalables y personalizadas.

a) Raspberry Pi y Python: La Raspberry Pi es una microcomputadora de bajo costo y tamaño compacto, ampliamente utilizada en aplicaciones de agricultura de precisión debido a su capacidad para ejecutar sistemas operativos completos (como Raspbian) y gestionar múltiples interfaces de entrada/salida. Python es el lenguaje de programación predeterminado en este entorno, y su flexibilidad permite implementar desde sistemas de adquisición de datos hasta plataformas completas de visualización y análisis.

Entre las bibliotecas más relevantes destacan:

- **RPi.GPIO y gpiozero:** Permiten la interacción directa con los pines GPIO, facilitando la lectura de sensores ambientales.
- **OpenCV y TensorFlow:** Para visión artificial en la identificación de enfermedades o el monitoreo del crecimiento vegetal.
- **Flask y FastAPI:** Permiten construir interfaces gráficas y APIs para monitoreo remoto.

Ejemplo aplicado: En Países Bajos, se han implementado invernaderos inteligentes basados en Raspberry Pi, los cuales ajustan automáticamente la temperatura y humedad con base en datos recopilados por sensores conectados a la plataforma (Carvalho et al., 2017).

b) ESP8266MOD y MicroPython: El ESP8266MOD es un módulo basado en el microcontrolador ESP8266 fabricado por Espressif Systems, que cuenta con conectividad Wi-Fi y un microprocesador de 32 bits con arquitectura Tensilica. Este dispositivo es ampliamente

utilizado para desarrollar aplicaciones IoT debido a su bajo consumo energético y facilidad de integración con sensores y actuadores (Espressif Systems, 2017).

Python, un lenguaje de programación ampliamente conocido por su simplicidad y versatilidad, puede ser utilizado en el ESP8266 a través de MicroPython. MicroPython es una implementación ligera de Python diseñada específicamente para microcontroladores con recursos limitados, permitiendo escribir scripts en Python para controlar hardware embebido (Damien George, 2014).

c) Arduino y Python: Arduino es una plataforma de prototipado electrónico de bajo costo que, aunque nativamente programada en C++, puede integrarse con Python mediante bibliotecas como PySerial. Esta combinación permite conectar el microcontrolador con otros sistemas (como Raspberry Pi o PC) para adquirir datos, automatizar procesos o transmitir información.

Aplicaciones comunes:

- Adquisición de datos de sensores de humedad, temperatura o radiación solar.
- Automatización del riego o sistemas de fertiirrigación.
- Comunicación mediante protocolos MQTT o HTTP hacia plataformas en la nube.

Ejemplo aplicado: En la India, se han desarrollado sistemas de riego autónomos con Arduino y Python que optimizan el consumo de agua mediante análisis de datos en tiempo real (Ranveer et al., 2018).

Tabla 1.

Comparativa del Uso de Python con Rasberry Pi, Arduino y ESP8266MOD en Agricultura Inteligente:

Características	Raspberry Pi	Arduino	ESP8266MOD	Python en Aplicación Agrícola
Tipo de Dispositivo	Microcomputadora	Microcontrolador	Microcontrolador con Wi-Fi	Ambos Permiten lectura de sensores y automatización
Lenguaje Nativo	Python	C++	MicroPython (Python reducido)	Python puede integrarse fácilmente con ambos.
Bibliotecas comunes	RPi.GPIO, OpenCV Flask	PySerial, MQTT	Umqtt.simple, Network, Machine	Adquisición de datos, visión artificial, control remoto.
Capacidad de Procesamiento	Alta (procesador ARM)	Baja (8 – 16 bits)	Media – Baja (32-bit tensilica)	Raspberry Pi permite ejecutar IA localmente.
Conectividad	Ethernet, Wi-Fi (USB)	Limitada (sin módulo externo)	Wi – Fi integrado	Facilita envío de datos a la nube o servidores.
Consumo energético	Alto	Bajo	Muy Bajo	Adecuado para dispositivos de campo con batería.
Aplicaciones destacadas	Invernaderos Inteligentes, Visión por computadora.	Riego automático, monitoreo básico	Monitoreo y control remoto de sensores, automatización Wi-Fi	Ambos contribuyen a una agricultura más eficiente.

Aplicación de Inteligencia Artificial en la Agricultura Automatizada mediante Python.

La incorporación de algoritmos de inteligencia artificial (IA) en la agricultura de precisión ha permitido una mejora significativa en la toma de decisiones automatizada, especialmente en aspectos como la predicción de condiciones climáticas, la detección de deficiencias nutricionales del suelo, el control de plagas, y la optimización del riego y otros recursos (Kamilaris & Prenafeta-Boldú, 2018). Python se ha consolidado como el lenguaje preferido para el desarrollo de soluciones basadas en IA, principalmente por su sintaxis clara y el vasto ecosistema de librerías especializadas (Van Rossum & Drake, 2009).

Entre las bibliotecas más destacadas se encuentran:

a. Tensor Flow

Desarrollada por Google, TensorFlow es una librería de código abierto diseñada para el desarrollo y entrenamiento de modelos de aprendizaje automático y redes neuronales profundas. En el contexto agrícola, ha sido utilizada para la clasificación automática de imágenes satelitales o de drones, la detección de enfermedades en cultivos mediante visión computacional y la predicción del rendimiento de cosechas (Dhingra et al., 2019).

b. Scikit-learn

Scikit-learn es una librería centrada en algoritmos de aprendizaje automático clásicos, como regresión logística, árboles de decisión, máquinas de soporte vectorial (SVM) y clustering. En agricultura, permite modelar datos

climáticos y edáficos, así como generar modelos predictivos para el consumo hídrico o la respuesta de los cultivos a distintas condiciones (Zhou et al., 2017).

c. Pytorch

promovida por Meta AI, es conocida por su flexibilidad y capacidad de entrenamiento dinámico. En entornos agrícolas se ha utilizado para desarrollar modelos de segmentación de imágenes, reconocimiento de enfermedades en cultivos y toma de decisiones autónoma basada en aprendizaje por refuerzo (Zhang et al., 2021).

En el siguiente ejemplo se representa un esquema básico de aprendizaje por esfuerzo. Se modela un agente que toma decisiones de riego en función del estado ambiental del cultivo.

d. Otras librerías complementarias

Librerías como **Keras** (una API de alto nivel que opera sobre TensorFlow), **OpenCV** (para visión por computadora), y **XGBoost** o **LightGBM** (para modelos de predicción de alto rendimiento) se utilizan frecuentemente como soporte en proyectos agrícolas automatizados. Además, herramientas como **Pandas** y **NumPy** resultan indispensables para la manipulación de grandes volúmenes de datos generados por sensores IoT y sistemas de monitoreo ambiental (Raschka & Mirjalili, 2019).

En conjunto, estas herramientas permiten la creación de sistemas inteligentes capaces no solo de interpretar datos, sino también de actuar de forma autónoma o semiautónoma. Esto se traduce en la optimización de recursos y la mejora de la productividad agrícola, con soluciones

que pueden implementarse tanto en la nube como en dispositivos Edge directamente en el campo.

Librerías de Python para la Automatización de Cultivos

Python destaca por su ecosistema de librerías que facilitan el desarrollo de sistemas automatizados, desde la adquisición de datos hasta la visualización y control remoto. La siguiente tabla resume las principales librerías utilizadas:

Tabla 2.

Librerías de Python para la automatización de cultivos

Categoría	Librería	Función Principal	Ejemplo de Uso
Control de Hardware	RPi.GPIO	Control de pines GPIO en Raspberry Pi	Lectura de sensor de humedad
Control de Hardware	PySerial	Comunicación serie con microcontroladores	Activación de bomba vía Arduino
Control de Hardware	PyModbus	Comunicación industrial (PLCs)	Control de válvulas Modbus en riego
Comunicación IoT	paho-mqtt	Transmisión de datos a la nube	Envío de datos a plataforma IoT
Comunicación IoT	requests	Peticiones HTTP REST	Envío de datos a una API web
Comunicación IoT (MicroPy)	urequests	Peticiones HTTP ligeras en microcontroladores	Envío de lectura de sensores a ThingSpeak
IA y Machine Learning	Scikit-learn	Modelos clásicos de ML	Predicción de necesidades de riego
IA y Machine Learning	TensorFlow	Redes neuronales profundas	Clasificación de enfermedades de cultivos
Análisis de Datos	Pandas	Manejo de datos tabulares	Análisis histórico de humedad
Análisis de Datos	NumPy	Cálculo numérico	Preprocesamiento de señales de sensores

Análisis de Datos	Statsmodels	Modelo estadístico y análisis predictivo	Regresión para predicción de humedad y temperatura
Visualización	Matplotlib	Gráficos estáticos	Gráfico de humedad del suelo
Visualización	Dash	Dashboards interactivos	Panel de monitoreo agrícola
Aplicaciones Web	Flask	Desarrollo web ligero	Visualización de datos en navegador

Arquitectura tecnológica de Python en cultivos inteligentes

La arquitectura tecnológica de un sistema de cultivos inteligentes basado en Python se estructura en diversas capas funcionales que interactúan para automatizar y optimizar la producción agrícola. Esta arquitectura se organiza en cinco niveles esenciales:

a. **Capa de Sensores y Actuadores**

Encargada de capturar datos físicos del entorno agrícola (como suelo, clima y plantas) y de ejecutar acciones automatizadas como el riego o la iluminación. Python interactúa con esta capa mediante librerías como RPi.GPIO, gpiozero, PySerial y PyModbus (Gupta et al., 2021).

b. **Capa de Control y Procesamiento Local (Edge Computing)**

Realiza el procesamiento inicial de datos en el mismo entorno agrícola, reduciendo la latencia y mejorando la eficiencia operativa. Se emplean dispositivos como Raspberry Pi y Jetson Nano, y librerías como NumPy, Pandas, Scikit-learn y TensorFlow Lite (Shi et al., 2016).

c. **Capa de Comunicación y Transporte de Datos**

Esta capa permite la transmisión de datos hacia plataformas en la nube mediante protocolos como MQTT (con la librería paho-mqtt) y HTTP (usando requests). Es fundamental para la sincronización y control remoto del sistema (Botta et al., 2016).

d. **Capa de Almacenamiento y Análisis de Datos**

Permite conservar datos históricos y aplicar análisis predictivos. Python ofrece compatibilidad con bases de datos como SQLite, InfluxDB y PostgreSQL, y herramientas de análisis como Pandas, Matplotlib, Scikit-learn y XGBoost para detectar patrones y generar recomendaciones (Wolfert et al., 2017).

e. **Capa de Visualización y Control Remoto**

Es la interfaz entre el usuario y el sistema, usualmente mediante aplicaciones web construidas con Flask, Dash o FastAPI. Estas permiten supervisar el estado del cultivo, visualizar métricas y ejecutar órdenes a distancia (Zhang et al., 2020).

Revisión del estado del arte del uso de Python para el monitoreo y control de cultivos automatizados con IoT

La integración de tecnologías emergentes en el ámbito agrícola ha propiciado un cambio significativo en la forma de producir alimentos. El uso de Python, en conjunto con dispositivos IoT, ha dado lugar al desarrollo de sistemas automatizados capaces de monitorear, controlar y predecir las condiciones agrícolas con una eficiencia sin precedentes. Esta sección explora los desarrollos más relevantes y actuales, enfocándose en aplicaciones concretas, tendencias emergentes, desafíos y oportunidades.

Aplicaciones actuales en la agricultura inteligente

Python es ampliamente adoptado en el desarrollo de soluciones para agricultura inteligente gracias a su compatibilidad con hardware de bajo costo y su extenso ecosistema de librerías. Entre sus principales aplicaciones destacan:

- Monitoreo ambiental

Consiste en la recolección de datos en tiempo real sobre variables como temperatura, humedad, radiación solar y pH del suelo mediante sensores conectados a dispositivos IoT. Python permite almacenar y analizar estos datos usando librerías como pandas, matplotlib, y numpy, a través de la conexión de los sensores a microcontroladores como Raspberry Pi y Arduino. Python así mismo utiliza librerías como PySerial, RPi.GPIO y Pandas para la adquisición y análisis de datos en tiempo real (Gupta et al., 2021).

- Automatización del riego

Sistemas que controlan el flujo de agua automáticamente con base en la lectura de sensores. Python implementa este control a través de librerías como PyModbus para la integración con controladores industriales, PySerial para la comunicación con microcontroladores, RPi.GPIO para activar electroválvulas y frameworks web como Flask (Ranveer et al., 2018).

- Procesamiento de imágenes

Para la visión artificial de cultivos se emplean cámaras conectadas a Rasberry Pi o NVIDIA Jetson programadas mediante librerías como OpenCV y TensorFlow, de esta forma Python permite detectar enfermedades, malezas o deficiencias nutricionales mediante visión artificial (Carvalho et al., 2017).

- Interacción con plataformas en la nube

Se realiza mediante el uso de protocolos MQTT (librería paho-mqtt) y HTTP (librería requests), lo cual facilita la sincronización y control remoto del sistema agrícola con plataformas como AWS IoT, Google Cloud o Azure.

Apartado Técnico de implementación

Ejemplo técnico: Monitoreo ambiental con Rasberry Pi y Python

- Hardware:
  - Rasberry Pi 4
  - Sensor DHT22 (temperatura y humedad)
  - Sensor de humedad de suelo capacitivo
  - Módulo Wifi integrado
- Software:
  - Python 3.10
  - Librerías: Adafruit_DHT, RPi.GPIO, pandas, matplotlib, paho-mqtt
- Funcionamiento básico del script:

```
import Adafruit_DHT
import time
sensor = Adafruit_DHT.DHT22
pin = 4  # GPIO conectado al sensor


while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
```

```
    if humidity is not None and temperature is not None:
        print(f"Temp: {temperature:.1f}°C, Humidity: {humidity:.1f}%")
    else:
        print("Sensor failure. Check wiring.")
    time.sleep(60)  # espera un minuto
```

Este scrip puede conectarse a la nube para transmitir los datos usando MQTT, lo que permite una visualización remota en tiempo real.

Casos de estudio relevantes

Caso 1- India: Sistema automatico de riego inteligente con Python y Arduino

Un proyecto implementado en campos de arroz en India integró sensores de humedad del suelo con un microcontrolador Arduino, gestionado por una Raspberry Pi corriendo scripts en Python. Este sistema redujo el uso de agua en un 30% y mejoró la eficiencia del riego en un 40% (Ranveer et al., 2018).

Caso 2 – Países Bajos: Invernaderos automatizados

Invernaderos en Países Bajos han sido automatizados con Raspberry Pi y Python, utilizando visión artificial con OpenCV para evaluar el estado de los cultivos. Sensores de temperatura y humedad ajustan automáticamente la ventilación y el riego. Este enfoque mejoró la tasa de crecimiento de las plantas y optimizó el uso energético en un 20% (Carvalho et al., 2017).

Análisis de resultados

Los estudios muestran que el uso de Python en sistemas agrícolas IoT tiene impactos positivos significativos:

Tabla 3.

Análisis de resultados

Métrica	**Promedio de mejora observada**
Reducción en el uso de agua[1]	20–35%
Aumento de la eficiencia energética[2]	10–25%
Mejora en la precisión del monitoreo[3]	30–50%
Disminución en pérdidas por enfermedades[4]	15–40%

Tendencias recientes del uso de Python para el monitoreo de cultivos automatizados con IoT

El ecosistema agrícola moderno ha comenzado a incorporar tecnologías emergentes que permiten no solo el monitoreo en tiempo real, sino también la toma de decisiones autónoma basada en datos. En este contexto, Python se ha posicionado como un lenguaje clave por su compatibilidad con plataformas IoT, su capacidad de procesamiento de datos y su integración con herramientas de inteligencia artificial. Las siguientes tendencias marcan el rumbo actual de su uso en cultivos automatizados:

- Implementación de modelos de IA en el borde (Edge AI): Gracias a bibliotecas como TensorFlow Lite, ONNX o MediaPipe, ahora es posible ejecutar modelos de machine learning directamente en microcontroladores como Raspberry Pi o

1 Ranveer et al. (2018): Sistema inteligente de riego en la India con Arduino y Python redujo el consumo de agua hasta en un 30%.

2 Carvalho et al. (2017): Automatización de invernaderos en Países Bajos logró una mejora del 20% en eficiencia energética.

3 Wolfert et al. (2017): Sistemas de monitoreo inteligente permiten reducir errores humanos y mejorar la precisión del monitoreo entre 30% y 50%.

4 Carvalho et al. (2017): Uso de visión artificial en Python detectó enfermedades tempranas, disminuyendo pérdidas en un 15–40%.

Jetson Nano, sin depender totalmente de la nube. Python facilita esta implementación mediante scripts optimizados para ejecutar modelos entrenados que pueden, por ejemplo, detectar plagas en imágenes captadas por cámaras o predecir enfermedades foliares. (Wolfert et al., 2017).

Ejemplo: Investigadores de la Universidad de Wageningen han utilizado Python y TensorFlow Lite en Raspberry Pi para detectar infecciones de mildiu en hojas de uva a través de visión por computadora (Wageningen UR, 2023).

- Integración de Python con plataformas IoT industriales: Plataformas como AWS IoT Core, Azure IoT Hub y Google Cloud IoT ofrecen SDKs y APIs compatibles con Python, permitiendo enviar datos desde sensores, ejecutar funciones Lambda o activar alertas. Esta integración acelera la adopción de soluciones comerciales escalables, con dashboards interactivos desarrollados en Plotly Dash, Streamlit o Flask.

Ejemplo: Startups en India están utilizando Python en módulos ESP32 conectados a sensores de humedad para enviar lecturas a la nube de Azure y activar alertas automáticas cuando los niveles de agua bajan del umbral (Patel et al., 2022).

- Agricultura de precisión mediante análisis de datos en tiempo real: La tendencia actual apunta hacia el análisis en tiempo real con Python mediante librerías como Pandas, NumPy, Matplotlib, y Scikit-learn. Esto permite crear modelos predictivos que evalúan factores como la evapotranspiración, necesidades de fertilización y predicción de cosechas.

Ejemplo: En Chile, se han implementado sistemas que miden variables climáticas cada 10 minutos y aplican modelos de regresión logística en Python para

predecir la mejor hora del día para regar viñedos, logrando un ahorro de agua del 30% (González et al., 2023).

- Automatización con Python y controladores industriales: Python se ha integrado con controladores industriales mediante protocolos como Modbus, OPC-UA y MQTT, permitiendo automatizar válvulas de riego, controlar el clima en invernaderos o activar dispensadores de nutrientes. Bibliotecas como pymodbus o paho-mqtt hacen esta conexión viable incluso para usuarios no especializados.
- Uso de gemelos digitales agrícolas: Los gemelos digitales son representaciones virtuales de cultivos o fincas completas. Python, en combinación con simuladores y librerías de modelado como SimPy, permite generar entornos virtuales donde se simulan condiciones climáticas, decisiones de riego y respuestas de las plantas.

Ejemplo: El proyecto europeo AI4FoodSecurity ha desarrollado gemelos digitales de cultivos de maíz y trigo que usan Python para evaluar escenarios de sequía o cambios bruscos de temperatura.

Tabla 4.

Tendencias recientes del uso de Python en agricultura inteligente.

Tendencia	**Descripción**	**Herramientas Python Involucradas**	**Beneficio Principal**
Edge AI	Modelos ML en el dispositivo	TensorFlow Lite, OpenCV	Reducción de latencia
Integración con la nube	Envío/recepción de datos	AWS SDK, Azure SDK, Paho-MQTT	Escalabilidad y monitoreo remote
Análisis en tiempo real	Procesamiento de datos de sensores	Pandas, Matplotlib, Scikit-learn	Decisiones basadas en datos
Automatización	Control de actuadores	PyModbus, PySerial	Ahorro de recursos y eficiencia
Gemelos digitales	Simulación de cultivos	SimPy, Matplotlib, NumPy	Planeación y predicción

Desafíos y oportunidades del uso de Python en cultivos automatizados con IoT

El uso de Python en la agricultura inteligente ha revolucionado la forma en que se gestionan los cultivos, no solo por su sintaxis accesible y su potente ecosistema de bibliotecas, sino también por su capacidad para integrarse con tecnologías IoT, inteligencia artificial y sistemas en la nube. Sin embargo, la adopción de esta tecnología presenta tanto oportunidades prometedoras como desafíos técnicos, logísticos y económicos.

Desafíos principales:

a. **Limitaciones de Infraestructura rural:** Una de las barreras más importantes para la implementación de soluciones IoT con Python es la conectividad. En muchas zonas rurales, el acceso a Internet es limitado o inestable, lo que impide el envío

continuo de datos a plataformas en la nube. Esta situación limita la operatividad de soluciones basadas en análisis en tiempo real o supervisión remota (Botta et al., 2016).

b. **Capacidad Técnica del personal agrícola:** El uso de Python y plataformas IoT requiere conocimientos básicos de programación, redes y manejo de sensores. En la mayoría de los casos, los agricultores no cuentan con la capacitación técnica necesaria, lo que genera dependencia de terceros para el mantenimiento y la expansión del sistema (Jayashree et al., 2018).

c. **Escalabilidad y mantenimiento de sistemas:** Si bien Python facilita el desarrollo de prototipos, escalar estas soluciones a grandes superficies agrícolas exige una arquitectura robusta de software, protocolos de comunicación estandarizados y sistemas de actualización remota. Esto implica una complejidad adicional en el diseño y el mantenimiento de sistemas distribuidos (Wolfert et al., 2017).

d. **Seguridad y privacidad de los datos:** La transmisión de datos agrícolas (clima, suelo, rendimiento, ubicación) requiere medidas de seguridad como cifrado, autenticación y control de acceso. Muchos sistemas prototipo desarrollados en Python omiten estas prácticas, exponiendo datos sensibles a vulnerabilidades (Botta et al., 2016; Shi et al., 2016).

Oportunidades de mejora y expansión

a. **Democratización del desarrollo tecnológico:** Python, al ser de código abierto y contar con una gran comunidad, permite a universidades, instituciones y desarrolladores independientes crear soluciones de bajo costo y adaptadas a contextos locales. Esto impulsa la innovación inclusiva en regiones agrícolas con recursos limitados (Zhang et al., 2020).

b. Integración con inteligencia artificial y análisis predictivo: El crecimiento de herramientas como Node-RED, ThingsBoard y MQTT, todas compatibles con Python, está permitiendo la creación de ecosistemas agrícolas modulares, escalables e interoperables. Esto facilita la integración de sensores, controladores y servicios de visualización sin depender de proveedores cerrados (Zhang et al., 2020).

c. Mejora de la sostenibilidad agrícola: Las soluciones basadas en Python permiten medir y reducir el uso de agua, energía y fertilizantes. Esta eficiencia operativa contribuye directamente a una producción más sostenible y resiliente ante el cambio climático (Wolfert et al., 2017).

Tabla 5.

Desafíos y Oportunidades de Uso de Python en la agricultura Inteligente

Categoría	**Desafíos principales**	**Oportunidades Emergentes**
Técnica	Limitaciones de hardware, interoperabilidad compleja	Bibliotecas optimizadas, soporte para multiples sensores y plataformas.
Humana	Escasez de capacitación técnica en Python.	Formación técnica online gratuita y crecimiento de comunidades open source.
Económica	Costos de hardware y servicios en la nube.	Software gratuito y plataformas comunitarias de bajo costo.
Infraestructura	Conectividad limitada, falta de redes de sensores robustas.	Edge computing, redes LoRaWAN y microservicios para zonas rurales.
Ambiental	Uso ineficiente de agua y recursos naturales.	Algoritmos en Python para gestión sostenible y predicción de riego.

Proyectos y Aplicaciones en Colombia que usan Python e IoT en el monitoreo de cultivos.

En Colombia, la implementación de tecnologías IoT en la agricultura ha ido en aumento, con varios proyectos que integran Python para el monitoreo y control de cultivos. A continuación, se presentan algunos de los desarrollos más relevantes en el país, los cuales reflejan el avance hacia una agricultura más eficiente, sostenible y tecnificada.

- Universidad Piloto de Colombia: Desarrolló un sistema de monitoreo y control para cultivos de fresa en Guasca (Cundinamarca), utilizando sensores de humedad y temperatura, integrados mediante IoT y controlados con Python para condiciones semi-hidropónicas (Universidad Piloto de Colombia, 2022).
- Universidad Industrial de Santander (UIS): Diseñó una plataforma IoT para el cultivo de tomate cherry en invernadero, monitoreando temperatura y humedad del aire mediante sensores, con procesamiento de datos en Python (UIS, 2023).
- Universidad El Bosque: Implementó un cultivo vertical automatizado con sensores de humedad, temperatura y control de nivel de agua, integrados en una interfaz web desarrollada en Python para el monitoreo remoto (Universidad El Bosque, 2022).
- Universidad Distrital Francisco José de Caldas: Diseñó un sistema de monitoreo agrícola utilizando tecnología LoRa para la transmisión de datos a larga distancia, integrando scripts en Python para la adquisición y análisis local de los datos (Universidad Distrital, 2022).
- Universidad Cooperativa de Colombia (Montería): Desarrolló una plataforma IoT para monitorear cultivos hidropónicos, integrando sensores de pH,

temperatura y conductividad eléctrica con visualización en tiempo real mediante Python (UCC, 2022).

- Proyecto independiente en GitHub: Un proyecto abierto colombiano propone el uso de visión artificial e inteligencia artificial con Python para identificar plagas y condiciones adversas en cultivos mediante cámaras y sensores conectados (Rojas, 2023).

Tabla 6.

Comparativa de Proyectos nacionales con Python e IoT

Proyecto / Institución	**Cultivo objetivo**	**Tecnologías utilizadas**	**Funciones implementadas**
Univ. Piloto de Colombia	Fresa	Python, sensores DHT, IoT	Monitoreo de clima y riego automático
UIS	Tomate Cherry	IoT, Python, sensores ambientales	Monitoreo en tiempo real en invernadero
Univ. El Bosque	Multicultivo vertical	Python, interfaz web, sensores	Visualización y automatización
Univ. Distrital	General agrícola	LoRa, Python	Monitoreo distribuido y bajo consumo
UCC Montería	Hidropónicos	Python, sensores pH/EC	Medición y control en tiempo real
GitHub / Independiente	General	Python, IA, visión artificial	Detección de plagas y análisis predictivo

Proyectos y Aplicaciones en Colombia que usan Python e IoT en el monitoreo de cultivos.

A nivel internacional, Python ha sido ampliamente adoptado en proyectos agrícolas impulsados por tecnologías de IoT. Desde países desarrollados como los Países Bajos y Japón hasta regiones agrícolas emergentes en Asia y África, la flexibilidad de Python permite su integración con sensores, sistemas de control y plataformas en la nube para optimizar cultivos. A continuación, se describen algunos proyectos representativos.

- Wageningen University & Research (Países Bajos): Desarrollo de invernaderos inteligentes automatizados usando Raspberry Pi y Python para ajustar temperatura, humedad y luz según algoritmos de aprendizaje automático (Carvalho et al., 2017).
- Mitsubishi Research Institute (Japón): Sistema de control de cultivos en interiores utilizando visión artificial con OpenCV y redes neuronales en Python para detectar enfermedades foliares (Kawashima et al., 2020).
- Digital Green (India): Implementación de plataformas Python-IoT en comunidades rurales para monitoreo del riego y alertas climáticas personalizadas (Gupta & Sharma, 2019).
- AgriTech Africa (Kenia): Sistema de monitoreo remoto de humedad del suelo con sensores de bajo costo y Python para predicción de sequías en cultivos de maíz (Okello et al., 2021).
- NASA Harvest (Estados Unidos): Plataforma de procesamiento de imágenes satelitales en Python para estimar rendimiento de cultivos y detectar zonas vulnerables (Lobell et al., 2022).

Tabla 7.

Comparativa de Proyectos Nacionales con Python e IoT

Institución / Proyecto	**País**	**Tecnologías utilizadas**	**Aplicación principal**
Wageningen UR	Países Bajos	Python, Raspberry Pi, sensores, ML	Automatización de invernaderos
Mitsubishi Research	Japón	Python, OpenCV, redes neuronales	Detección de enfermedades foliares
Digital Green	India	Python, IoT, sensores	Monitoreo del riego y alertas
AgriTech Africa	Kenia	Python, sensores de humedad, cloud	Predicción de sequía
NASA Harvest	EE.UU.	Python, imágenes satelitales, AI	Estimación de rendimiento agrícola

Librerías a Usar por el Proyecto

Tabla 8.

Librerías a Usar por el Proyecto

Entorno	Librería / Recurso	Función principal
ESP8266MOD	machine	Control de pines GPIO y periféricos del microcontrolador (MicroPython, 2023).
	dht	Lectura de temperatura y humedad desde sensores DHT11/DHT22 (MicroPython, 2023).
	network	Manejo de la conexión WiFi del ESP8266 (MicroPython, 2023).
	urequests	Envío de datos al servidor mediante solicitudes HTTP POST (MicroPython, 2023).
	time	Control de temporización en los ciclos de monitoreo (MicroPython, 2023).
Python (backend)	flask	Framework ligero para la creación de servicios web RESTful (Grinberg, 2018).
	csv	Lectura y escritura de archivos CSV para el almacenamiento de datos (Python Software Foundation, 2023).
	datetime	Generación de marcas de tiempo para los registros del sistema (Python Software Foundation, 2023).
	dash	Creación de dashboards web interactivos para visualización de datos (Plotly, 2023a).
	pandas	Manipulación y análisis de datos estructurados tipo tabla (McKinney, 2018).
	plotly	Visualización gráfica de datos en la web con gráficos interactivos (Plotly, 2023b).
Frontend (VS Code)	HTML/CSS/JavaScript	Estructura, estilos e interacción de la interfaz web (Mozilla Developer Network, 2024).
	Chart.js (opcional)	Representación de datos mediante gráficos dinámicos en el navegador (Chart.js Contributors, 2024).
	Bootstrap (opcional)	Diseño web responsivo y moderno mediante clases CSS predefinidas (Bootstrap Team, 2024).

Materiales y Métodos

Materiales

.

Para el desarrollo del sistema de monitoreo aeropónico se utilizaron los siguientes recursos tecnológicos y herramientas:

Tabla 9.

Hardware

Componente	**Descripción**
ESP8266MOD (NodeMCU)	Microcontrolador con conectividad WiFi, encargado de captar y transmitir datos.
Sensor DHT11	Sensor digital de temperatura y humedad de alta precisión.
Cables Dupont	Cables para conexión entre sensor y microcontrolador.
Protoboard (opcional)	Plataforma para ensamblaje de circuitos sin soldadura.
Fuente USB 5V	Alimentación para el ESP8266MOD desde el computador.
Computador portátil HP	Equipo principal de desarrollo y ejecución de backend y dashboard.

Tabla 10.

Software

Software / Herramienta	**Función**
MicroPython	Firmware ligero para programar el ESP8266MOD (MicroPython, 2023).
uPyCraft IDE	Entorno de desarrollo para cargar scripts en el microcontrolador usando MicroPython.
Python 3.10+	Lenguaje de programación principal del backend del sistema.
Flask	Framework para construir la API REST de recepción de datos (Grinberg, 2018).
Spyder IDE	Entorno de desarrollo científico usado para programar el backend.
Visual Studio Code	Editor para la interfaz frontend del dashboard web.
Dash	Librería para construir interfaces visuales en tiempo real (Plotly, 2023a).
Pandas y Plotly	Análisis y visualización de datos recolectados.

Metodología

Enfoque Metodológico

El presente proyecto aplicado adopta un enfoque cuantitativo y tecnológico, sustentado en el paradigma empírico-analítico, ya que se parte de variables observables (temperatura y humedad) que pueden ser medidas, procesadas y evaluadas mediante herramientas computacionales. El desarrollo del sistema responde a un modelo de investigación aplicada, cuyo propósito es resolver un problema concreto en un entorno real: el monitoreo de cultivos automatizados e inteligentes mediante tecnologías IoT y lenguajes de programación accesibles como Python.

Según Hernández et al. (2014), la investigación aplicada se orienta a "proponer soluciones prácticas a problemas específicos utilizando conocimientos científicos y tecnológicos" (p. 6). En concordancia con ello, este proyecto se apoya en una metodología de prototipado rápido, en la cual el sistema fue diseñado, implementado y probado en condiciones controladas, con el fin de validar la comunicación entre hardware, backend, frontend y la nube (Render como servicio de despliegue).

Asimismo, el estudio presenta un carácter descriptivo y exploratorio.

- **Descriptivo**, porque documenta la arquitectura del sistema, los componentes utilizados y los resultados de su funcionamiento.
- **Exploratorio**, porque se trata de un prototipo inicial que permite sentar las bases para futuras investigaciones y mejoras orientadas a la escalabilidad en contextos agrícolas más amplios.

Podemos decir que el enfoque metodológico combina un diseño experimental-prototípico con un carácter aplicado, descriptivo y exploratorio, fundamentado en la validación práctica de un sistema de monitoreo IoT.

Diseño Metodológico

El diseño metodológico del proyecto se estructuró en las siguientes etapas:

- **Análisis de requerimientos técnicos:** Se identificaron las necesidades de los cultivos automatizados en términos de monitoreo ambiental, definiendo como parámetros críticos la temperatura y la humedad relativa. Con base en ello, se seleccionaron los componentes de hardware (ESP8266MOD y sensor DHT11) y las herramientas de software (MicroPython, Flask, PostgreSQL y entornos de desarrollo).
- **Desarrollo del sistema:** Se implementó el prototipo utilizando el microcontrolador ESP8266MOD, programado en MicroPython, encargado de recolectar datos del sensor DHT11 y transmitirlos vía WiFi a la plataforma en la nube ThingSpeak. Desde allí, el backend en Python con Flask consume los datos, los procesa y los almacena en una base de datos PostgreSQL desplegada en Render. Paralelamente, se diseñó el frontend web (HTML, CSS y JavaScript), encargado de mostrar dashboards con gráficas dinámicas y alertas visuales sobre los parámetros recolectados.
- **Pruebas y validación funcional:** El sistema fue probado verificando principalmente la conexión y recepción de datos desde el sensor DHT11 hacia el microcontrolador, la transmisión de datos a ThingSpeak y la integración entre los diferentes entornos (backend en Flask, base de datos en PostgreSQL, frontend web, despliegue en Render y repositorio en GitHub). Estas pruebas permitieron confirmar la correcta comunicación y funcionamiento básico del prototipo.

- **Documentación técnica, análisis de resultados y despliegue:** La documentación se centró en la elaboración del documento de investigación del proyecto de grado aplicado, en el que se describe la fundamentación teórica, el diseño metodológico, el desarrollo del sistema, las pruebas realizadas y los resultados obtenidos. Asimismo, se incluyen reflexiones sobre la escalabilidad y posibles mejoras en el contexto de cultivos inteligentes.

Técnicas e Instrumentos

- **Instrumentos de medición:** sensor DHT11 para la obtención de datos de temperatura y humedad relativa.
- **Instrumentos computacionales**: microcontrolador ESP8266MOD, programación con MicroPython, desarrollo de API REST con Flask, almacenamiento en PostgreSQL y visualización de datos en el frontend web (HTML, CSS, JS y gráficas dinámicas con Plotly).

Desarrollo Del Proyecto

Fase de análisis de requerimientos

Se identificaron las necesidades operativas del entorno de cultivo aeropónico, estableciendo como parámetros clave de monitoreo la temperatura y la humedad relativa. Se seleccionó el microcontrolador ESP8266MOD por su bajo consumo energético, conectividad WiFi integrada y compatibilidad con MicroPython. Para la medición ambiental se escogió el sensor DHT11, por su precisión y bajo costo.

Para esto es necesario determinar la arquitectura del sistema, la cual consiste en cuatro capas:

1. **Capa de sensado (Dispositivo Fisico):** Sensor DHT11 que mide temperatura y humedad del entorno del cultivo inteligente. Y ESP8266MOD (NodeMCU) que es un microcontrolador que permite Lee los datos del DHT11, se conecta a la red WiFi y envía los datos al servidor backend mediante una solicitud HTTP POST.
2. **Capa de Red:** Consiste en WiFi doméstica o local que transmite los datos del ESP8266MOD al servidor ejecutado en una red local (o eventualmente en la nube). Los sensores de temperatura y humedad, conectados al microcontrolador ESP8266MOD, envían datos al servicio en la nube ThingSpeak, que actúa como una pasarela inicial de almacenamiento y consulta. Desde allí, el backend obtiene los registros más recientes a través de solicitudes HTTP a la API REST del canal configurado.
3. **Capa de Procesamiento** (Backend Python): El núcleo del backend se implementó con Flask siguiendo un enfoque **orientado a servicios** (REST API), donde cada funcionalidad se encapsula en rutas específicas, asegurando así la interoperabilidad con el frontend web y la posibilidad de integrarse en el futuro con aplicaciones móviles u otros servicios en la nube.

4. **Capa de Visualización (frontend/dashboard):** La fase de visualización de datos constituye un componente central en el sistema de monitoreo IoT, dado que transforma los registros obtenidos por los sensores en información comprensible, interactiva y útil para la toma de decisiones en el contexto de cultivos inteligentes. Usa **HTML, CSS y JavaScript**, con conexión a un backend implementado en **Flask (Python)**. Esta interfaz ofrece dos funcionalidades principales: (i) el monitoreo en tiempo real de la temperatura y la humedad, y (ii) la gestión de umbrales para configurar límites de alerta.

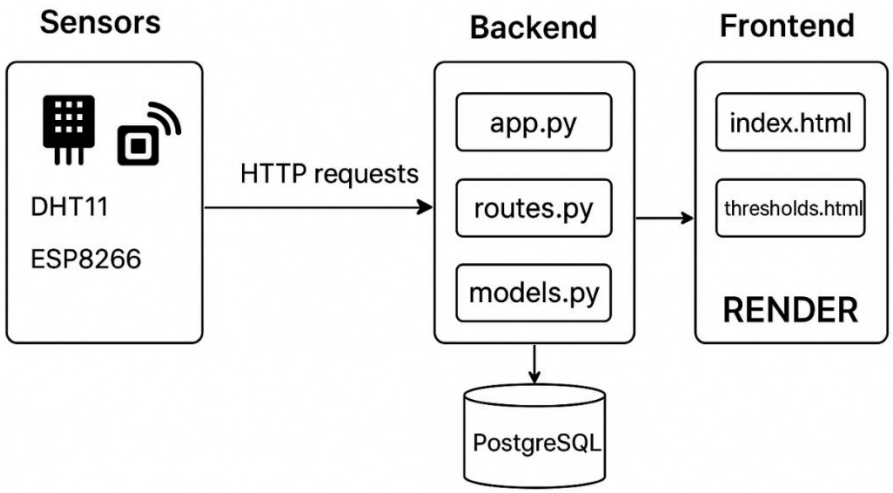
```
Sistema de Monitoreo/
├── backend/
│       └── app.py
│       └── config.py
│       └── routes.py
│       └── models.py
│       └── requirements.txt
│       └── database.db
└── frontend/
├── index.html
├── thresholds.html
└── assets /
        ├── css /
        │       └── styles.css
        └── js /
                └── api.js
```

└── dashboard.js

└── thresholds.js

Figura 7.

Diagrama de arquitectura del sistema



La arquitectura del sistema propuesto se fundamenta en un enfoque modular, escalable y desplegado en la nube, que garantiza la integración de datos en tiempo real provenientes del entorno agrícola con las herramientas de análisis y visualización desarrolladas.

En el contexto de este proyecto, los datos son generados por sensores DHT11 conectados a un microcontrolador ESP8266, encargado de la captura de variables ambientales como temperatura y humedad. Estos dispositivos transmiten la información a la plataforma ThingSpeak, la cual actúa como punto de integración IoT mediante APIs, centralizando los registros antes de ser procesados por la aplicación.

El backend, implementado en Python y desplegado en Render, utiliza el microframework Flask para el desarrollo de una API web que gestiona el flujo de datos y permite la interacción con el frontend. Se incorporó Flask-CORS para habilitar el acceso desde distintos orígenes y SQLAlchemy como ORM, lo que facilita la abstracción y manipulación de la base de datos. A diferencia de versiones locales iniciales que empleaban SQLite, la versión final del sistema utiliza PostgreSQL en la nube, garantizando un mayor rendimiento, persistencia y capacidad de escalabilidad.

La comunicación con servicios externos, particularmente con ThingSpeak, se logra mediante la librería requests, permitiendo la automatización en la transferencia de datos desde los sensores hacia el backend. Posteriormente, estos datos son almacenados en PostgreSQL y utilizados para alimentar los módulos de visualización y control.

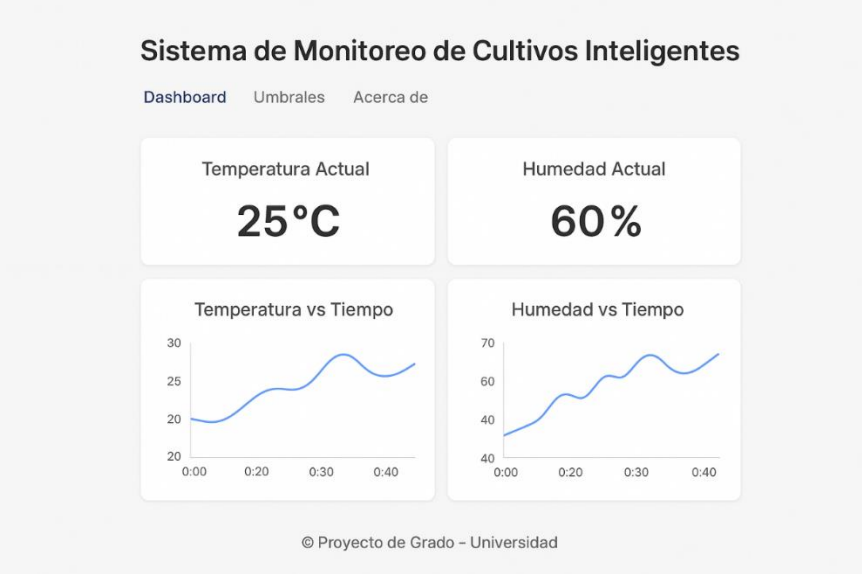
El frontend, compuesto por interfaces en HTML, CSS y JavaScript, permite a los usuarios visualizar los registros históricos, interpretar métricas de monitoreo y establecer umbrales de control. Los archivos api.js, dashboard.js y thresholds.js aseguran la comunicación con la API del backend, mientras que las vistas principales (index.html y thresholds.html) presentan la información de forma clara y accesible.

En conjunto, esta arquitectura refleja un ecosistema tecnológico en el cual la captura, transmisión, almacenamiento y visualización de datos se encuentran integrados. Al centralizar la información en tiempo real y automatizar el análisis, el sistema habilita la toma de decisiones basadas en datos, constituyendo una solución innovadora y efectiva para el monitoreo agrícola en entornos de producción inteligente.

Ahora se propone una maquetación o mockup de la posible visualización del sistema web de monitoreo.

Figura 8.

Mockup del sistema de monitoreo.



Fase de conexión de la red y los sensores

Durante esta fase se llevó a cabo la configuración inicial del microcontrolador ESP8266MOD (NodeMCU) empleando el firmware MicroPython y el entorno uPyCraft IDE**.** El código implementado permitió establecer la comunicación del sistema con la red y la captura de datos de los sensores.

En primer lugar, se realizó la configuración de red**,** mediante la cual el ESP8266MOD se conectó a la red WiFi local utilizando las credenciales definidas (SSID y contraseña). Posteriormente, se implementó la lectura del sensor DHT11**,** encargado de medir la temperatura y la humedad relativa a través del pin GPIO4 (D2) del microcontrolador.

Finalmente, se estableció la integración con la plataforma ThingSpeak**,** que actúa como repositorio intermedio de datos en la nube. Debido a que ThingSpeak restringe la frecuencia de

transmisión, el sistema fue configurado para enviar los registros en intervalos de 15 a 20 segundos, asegurando la compatibilidad y estabilidad en el flujo de datos.

Figura 9.

Código main.py

```
import network
import urequests
import time
import dht
from machine import Pin

# === CONFIG WiFi ===
SSID = "[redacted]"        # WiFi
PASSWORD = "[redacted]"

# === CONFIG ThingSpeak ===
API_KEY = "TU_API_KEY"  # Write API Key
THINGSPEAK_URL = "https://api.thingspeak.com/update?api_key=" + API_KEY

# === Configurar WiFi ===
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(SSID, PASSWORD)

print("Conectando a WiFi...")
while not wifi.isconnected():
    pass
print("Conectado a WiFi:", wifi.ifconfig())

# === Configurar DHT11 ===
sensor = dht.DHT11(Pin(4))  # GPIO4 = D2 en NodeMCU

# === Bucle principal ===
while True:
    try:
        sensor.measure()
        temp = sensor.temperature()
        hum = sensor.humidity()
```

Figura 10.

Código main.py

```

        print("Temperatura:", temp, "   Humedad:", hum, "%")

        # Enviar a ThingSpeak
        url = "{}&field1={}&field2={}".format(THINGSPEAK_URL, temp, hum)
        response = urequests.get(url)
        response.close()

        print("Datos enviados a ThingSpeak")

    except Exception as e:
        print("Error:", e)

    time.sleep(15)  # ThingSpeak requiere minimo 15 seg entre envios

```

Figura 11.

Prueba 1 lectura y conexión de sensores usando ESP8266

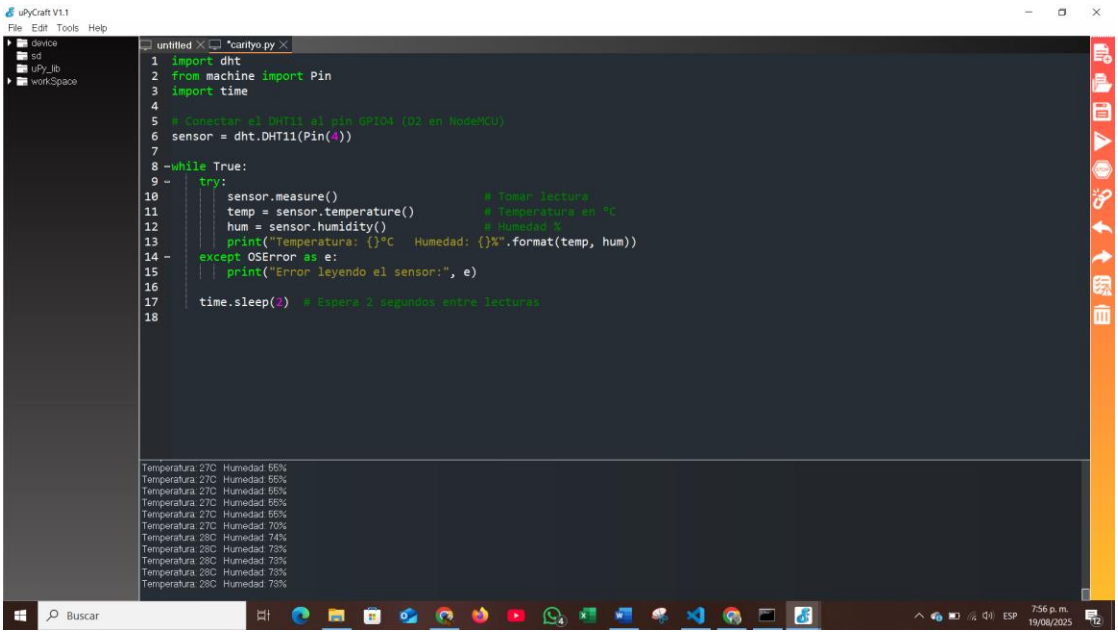


Figura 12.

Código de prueba de conexión entre uPyCraft y ThingSpeak con valores estáticos

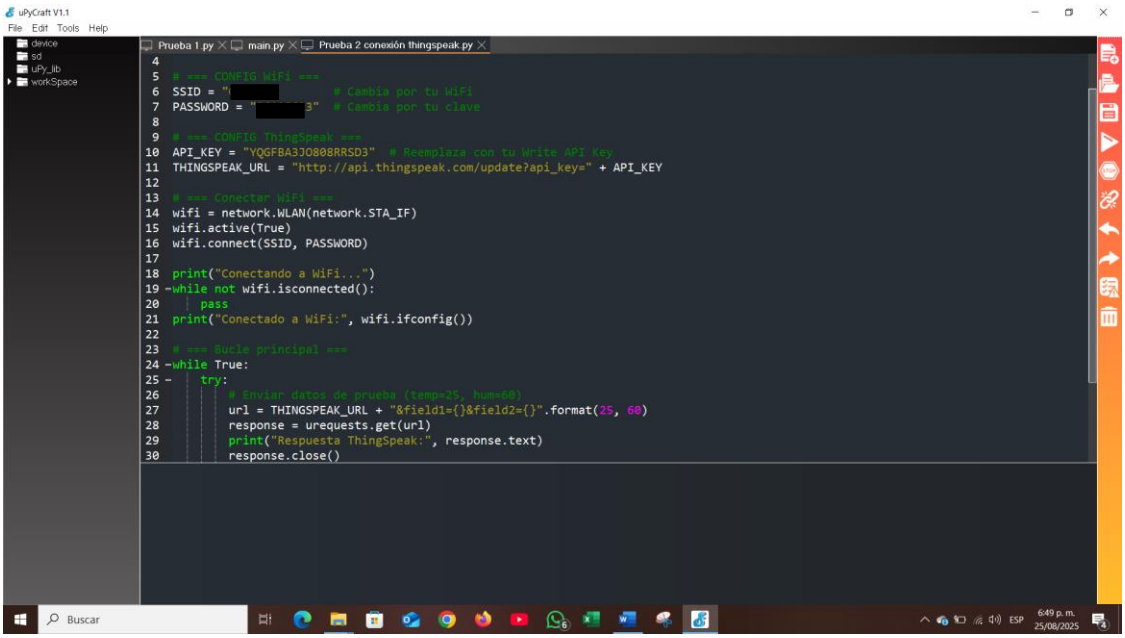


Figura 13.

Código Main de Conexión de sensores con aplicativo ThingSpeak

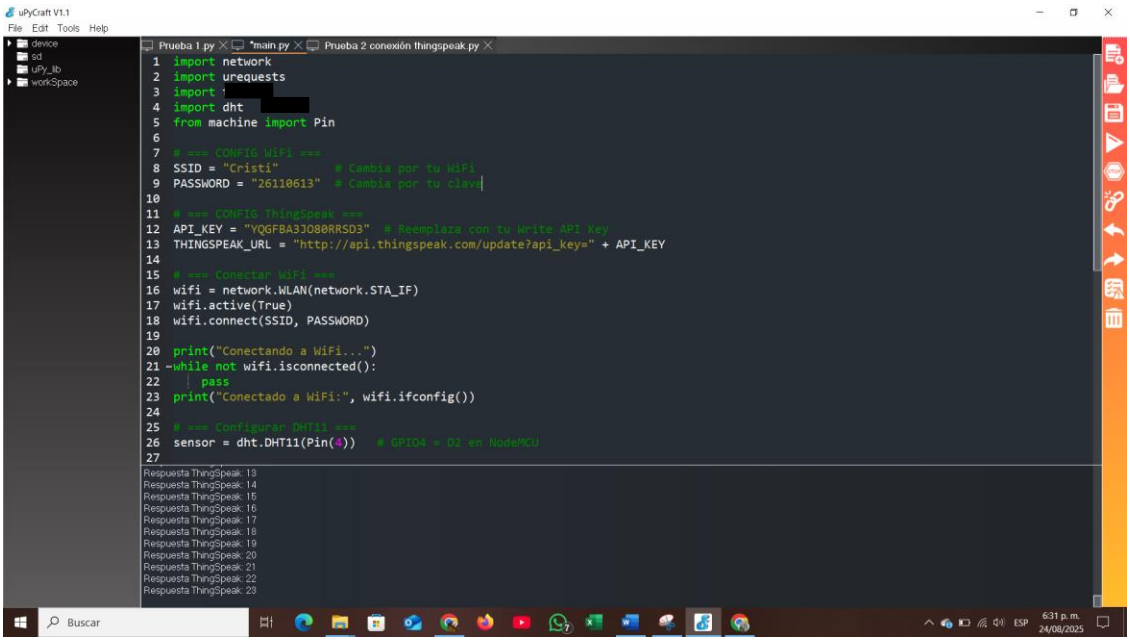


Figura 14.

Conexión de los sensores con ThingSpeak

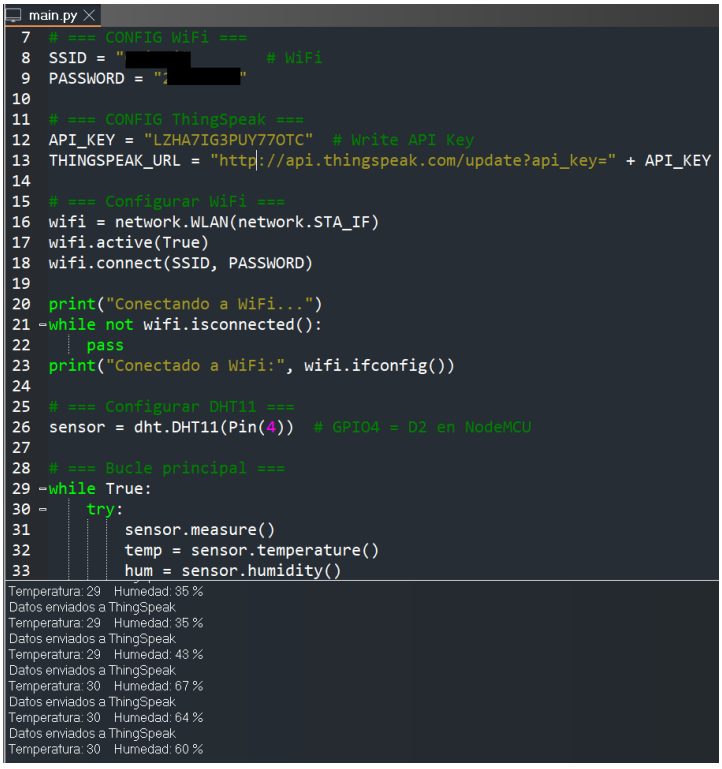


Figura 15.

Prueba 1 conexión de uPyCraft con ThingSpeak con variables fijas.

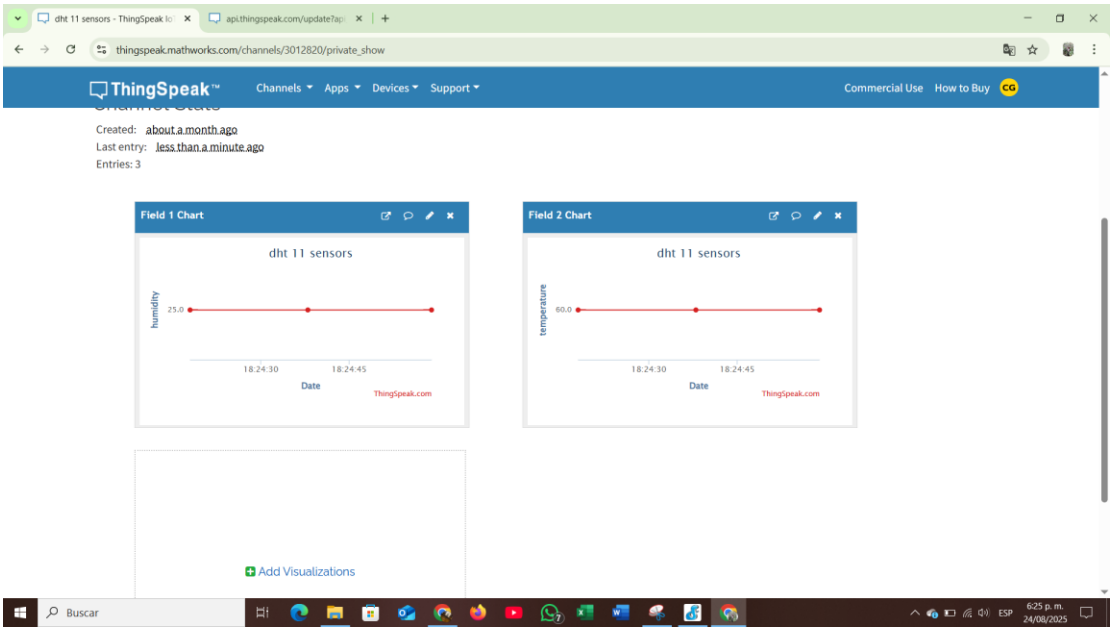
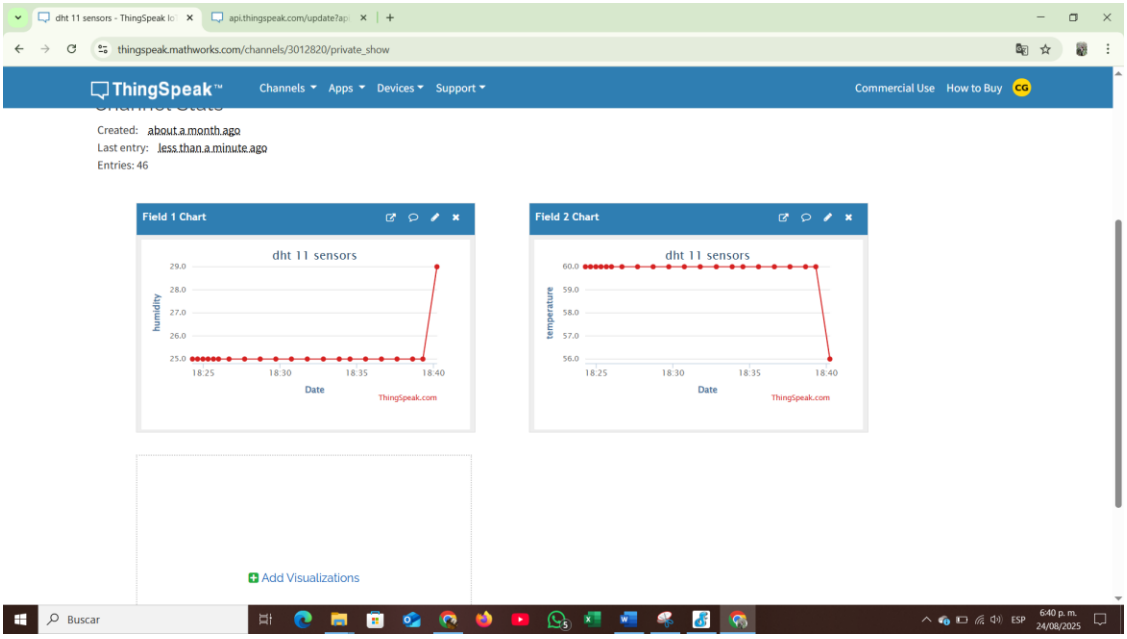


Figura 16.

Conexión del archivo main con ThingSpeak



Fase de desarrollo del backend del sistema

El backend constituye el núcleo encargado de gestionar los procesos de adquisición, almacenamiento, procesamiento y control de los datos provenientes de los sensores conectados a la red IoT. Esta capa fue desarrollada en Python utilizando el framework Flask**,** seleccionado por su ligereza, flexibilidad y compatibilidad con bibliotecas científicas y de análisis de datos.

El desarrollo del backend se organizó en módulos principales, garantizando la modularidad y escalabilidad del sistema:

- **Configuración del Sistema (config.py):** Este módulo centraliza las variables globales del backend, como la conexión con el servicio en la nube ThingSpeak y la cadena de conexión con la base de datos en PostgreSQL. Asimismo, define valores predeterminados de los umbrales de temperatura y humedad, evitando la dispersión de configuraciones en el código y facilitando futuras modificaciones.

Figura 17.

Código config.py

```
# config.py
from flask_sqlalchemy import SQLAlchemy

# Configuración de ThingSpeak
THINGSPEAK_CHANNEL_ID = "3012820"
THINGSPEAK_API_KEY = "J437OMJ3WVBPSLYV"
THINGSPEAK_URL = f"https://api.thingspeak.com/channels/{THIN

# Umbrales por defecto
DEFAULT_TEMP_MAX = 30
DEFAULT_TEMP_MIN = 15
DEFAULT_HUM_MAX = 80
DEFAULT_HUM_MIN = 30

# Instancia de SQLAlchemy
db = SQLAlchemy()
```

- **Definición de modelos de datos (models.py):** Para la gestión de la base de datos, se utilizó SQLAlchemy como ORM (Object Relational Mapping). Se establecieron dos modelos principales:
    - Umbrales: registra los valores mínimo y máximo de temperatura y humedad configurados para los cultivos.
    - Histórico: almacena los registros de temperatura, humedad y fecha de captura, conformando una bitácora que respalda la información transmitida desde la nube.

Figura 18.

Código models.py

```python
# models.py
from backend.config import db
class Umbrales(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    temp_min = db.Column(db.Float, default=15)
    temp_max = db.Column(db.Float, default=30)
    hum_min = db.Column(db.Float, default=30)
    hum_max = db.Column(db.Float, default=80)
class Historico(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    fecha = db.Column(db.String(50))
    temp = db.Column(db.Float)
    hum = db.Column(db.Float)
```

Figura 19.

Iniciación del modelo de tabla si no existen

```python
@app.before_request
def init_db():
    global _db_initialized
    if _db_initialized:
        return

    db.create_all()  # crear tablas si no existen
    db.session.commit()
    _db_initialized = True  # marcar como inicializado

# ---------- Ejecutar en local ----------
if __name__ == "__main__":
    with app.app_context():
        init_db()
    app.run(host="0.0.0.0", port=5000, debug=True)
```

- **Inicialización del servidor y la base de datos (app.py):** Este archivo constituye el punto de entrada del backend. Aquí se configura el servidor Flask, se habilitan las dependencias principales, se inicializa la conexión con la base de datos PostgreSQL y se registran los distintos módulos de rutas y modelos que conforman el sistema.

Figura 20.

Código archivo app.py

```
@author: User
"""
# app.py
from flask import Flask, send_from_directory
from flask_cors import CORS
from backend.config import db
from backend.routes import routes
import os

app = Flask(
    __name__,
    static_folder="../frontend",
    static_url_path="/"
)

# Configuración de la base de datos
app.config['SQLALCHEMY_DATABASE_URI'] = os.getenv(
    'DATABASE_URL', 'sqlite:///database.db'
)
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

CORS(app)
db.init_app(app)

# Registrar rutas
app.register_blueprint(routes, url_prefix="/api")

@app.route("/")
def index():
    return send_from_directory(app.static_folder, "index.html")

@app.route("/thresholds")
def thresholds():
    return send_from_directory(app.static_folder, "thresholds.html")

# ---------- Inicialización de la DB ----------
_db_initialized = False  # flag para correr solo una vez

@app.before_request
def init_db():
    global _db_initialized
    if _db_initialized:
        return

    db.create_all()  # crear tablas si no existen
    db.session.commit()
    _db_initialized = True  # marcar como inicializado

# ---------- Ejecutar en local ----------
if __name__ == "__main__":
    with app.app_context():
        init_db()
    app.run(host="0.0.0.0", port=5000, debug=True)
```

- **Gestión de rutas y lógicas de negocio (routes.py):** Se implementaron los **endpoints REST** que permiten la comunicación con el frontend y la interacción con los datos. Entre sus funciones principales se encuentran:
    - Recuperar registros desde la API de **ThingSpeak**.
    - Guardar los datos históricos en la base de datos PostgreSQL.
    - Consultar y actualizar los valores de umbrales definidos.
    - Validar los parámetros y generar alertas cuando se detectan valores fuera de los rangos establecidos

Figura 21.

Parte del código del archivo de routes.py

```python
# routes.py
from flask import Blueprint, jsonify, request
import requests
from backend.config import db
from backend.models import Umbrales, Historico
from backend.config import THINGSPEAK_URL

routes = Blueprint("routes", __name__)

# 🔷 Obtener datos de ThingSpeak
@routes.route("/thingspeak", methods=["GET"])
def obtener_datos_thingspeak():
    try:
        r = requests.get(THINGSPEAK_URL, timeout=5)
        data = r.json()
        feeds = data.get("feeds", [])
        datos = []
        for f in feeds:
            temp = float(f.get("field1") or 0)
            hum = float(f.get("field2") or 0)
            fecha = f["created_at"]

            # Guardar en histórico
            registro = Historico(fecha=fecha, temp=temp,
            db.session.add(registro)
        db.session.commit()

        # Formato para frontend
        datos = [
            {"fecha": f["created_at"], "temp": f.get("f
            for f in feeds
        ]
        return jsonify(datos)
    except Exception as e:
        return jsonify({"error": str(e)}), 500


# 🔷 Obtener histórico completo (desde SQLite)
@routes.route("/historico", methods=["GET"])
```

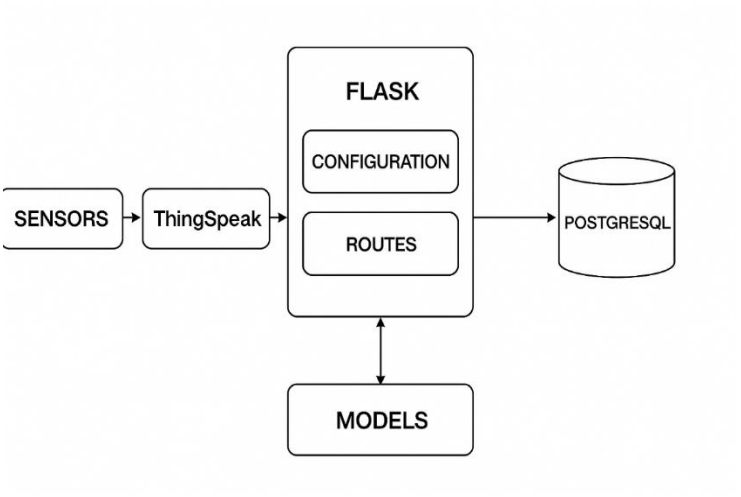
- **Base de datos SQLite:** Para el almacenamiento persistente en la nube se optó por **PostgreSQL**, un motor de base de datos robusto, confiable y ampliamente utilizado en aplicaciones web modernas. Su elección respondió a la necesidad de desplegar el sistema en **Render**, dado que esta plataforma no admite adecuadamente el uso de SQLite en producción. Con PostgreSQL se asegura:
    - Integridad transaccional de los datos.
    - Manejo eficiente de múltiples conexiones concurrentes.
    - Escalabilidad y persistencia en el tiempo de los registros históricos y los umbrales definidos.

En conjunto, el backend recibe periódicamente la información transmitida por los sensores a través de ThingSpeak**,** la procesa en Python y la almacena en la base de datos PostgreSQL. Estos datos se validan frente a los umbrales establecidos y, en caso de superar los límites configurados, se generan alertas que pueden ser consultadas desde el frontend mediante los endpoints definidos.

Este diseño modular garantiza que el backend funcione como la columna vertebral del sistema de monitoreo**,** asegurando la correcta integración entre los sensores, la nube y la base de datos en la nube. Además, la arquitectura planteada permite escalar el sistema en fases posteriores, con la posibilidad de incorporar algoritmos de predicción, sistemas de riego automatizado o analítica avanzada de datos agrícolas.

Figura 22.

Diagrama Lógico de arquitectura del Backend



Fase de visualización de datos

La fase de visualización constituye la capa de interacción directa entre el usuario y el sistema de monitoreo, cuyo objetivo principal es ofrecer una representación clara y en tiempo

real de los parámetros ambientales críticos (temperatura y humedad), así como la posibilidad de configurar umbrales de control. Esta fase se desarrolló mediante una interfaz web ligera construida con HTML, CSS y JavaScript, conectada a un backend en Flask **(**Python) que gestiona los datos recibidos desde ThingSpeak y la base de datos PostgreSQL.

Los componentes principales que integran esta capa son los siguientes:

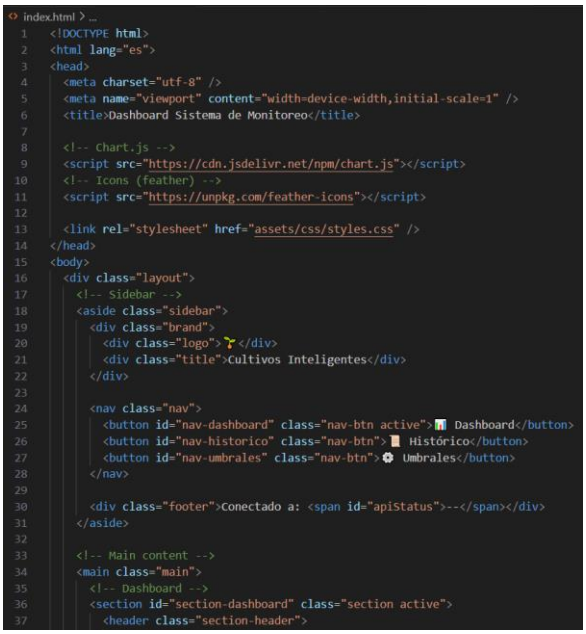
- **Diseño de la Interfaz Principal (index.html):** El archivo index.html constituye la página de inicio y la interfaz principal del sistema. Presenta al usuario, de manera estructurada y visual, los datos de temperatura y humedad recolectados por los sensores y transmitidos a la nube ThingSpeak, posteriormente gestionados por el backend en Flask La interfaz incluye:

  - Un encabezado de navegación básica.
  - Tarjetas con los valores actuales de temperatura y humedad.
  - Gráficos dinámicos para el análisis de tendencias.

El diseño prioriza claridad, simplicidad y usabilidad**,** garantizando que el usuario pueda interpretar rápidamente el estado del cultivo.

Figura 23.

Parte del código del index.html



```
index.html > ...
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Dashboard Sistema de Monitoreo</title>

  <!-- Chart.js -->
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <!-- Icons (feather) -->
  <script src="https://unpkg.com/feather-icons"></script>

  <link rel="stylesheet" href="assets/css/styles.css" />
</head>
<body>
  <div class="layout">
    <!-- Sidebar -->
    <aside class="sidebar">
      <div class="brand">
        <div class="logo">🌱</div>
        <div class="title">Cultivos Inteligentes</div>
      </div>

      <nav class="nav">
        <button id="nav-dashboard" class="nav-btn active">📊 Dashboard</button>
        <button id="nav-historico" class="nav-btn">📋 Histórico</button>
        <button id="nav-umbrales" class="nav-btn">⚙️ Umbrales</button>
      </nav>

      <div class="footer">Conectado a: <span id="apiStatus">--</span></div>
    </aside>

    <!-- Main content -->
    <main class="main">
      <!-- Dashboard -->
      <section id="section-dashboard" class="section active">
        <header class="section-header">
```

Figura 24.

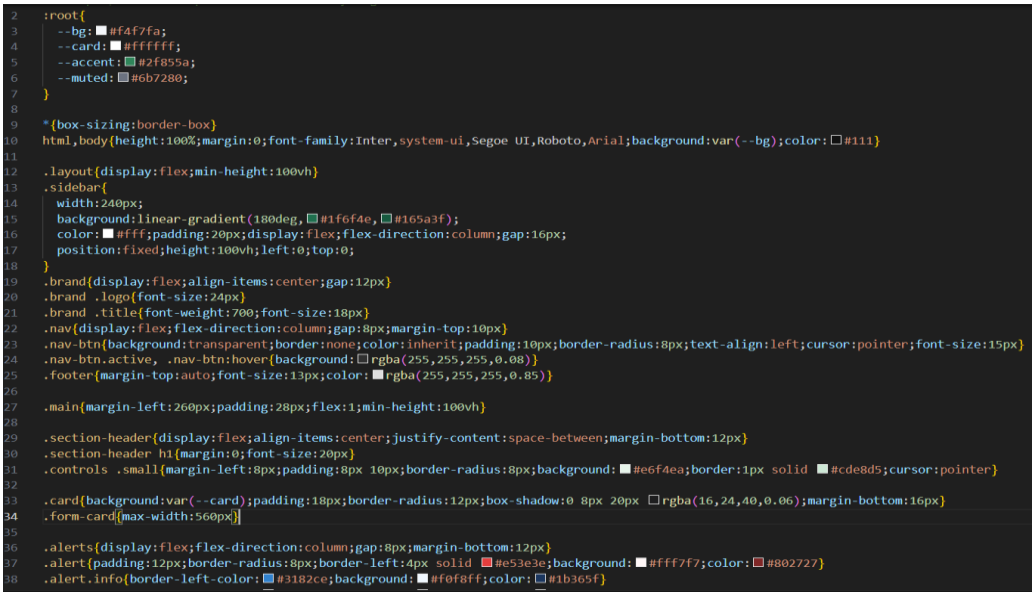
Parte del código del index

```
    <section id="section-umbrales" class="section">
      <header class="section-header"><h1>Umbrales</h1></header>
      <div class="card form-card">
        <form id="formThresholds">
          <div class="row">
            <label>Temperatura mínima (°C)</label>
            <input type="number" step="0.1" id="tempMin" required />
          </div>
          <div class="row">
            <label>Temperatura máxima (°C)</label>
            <input type="number" step="0.1" id="tempMax" required />
          </div>
          <div class="row">
            <label>Humedad mínima (%)</label>
            <input type="number" step="0.1" id="humMin" required />
          </div>
          <div class="row">
            <label>Humedad máxima (%)</label>
            <input type="number" step="0.1" id="humMax" required />
          </div>

          <div class="row actions">
            <button type="submit" class="primary">Guardar</button>
            <button type="button" id="btnReset" class="muted">Restablecer</button>
          </div>
        </form>
      </div>
    </section>
  </main>
</div>
```

- **Estilos y usabilidad (styles.css):** En este módulo se definieron las reglas de estilo para asegurar una visualización limpia, adaptable y responsiva en distintos dispositivos. Se buscó un diseño minimalista que facilite la interpretación de la información.

Figura 25.

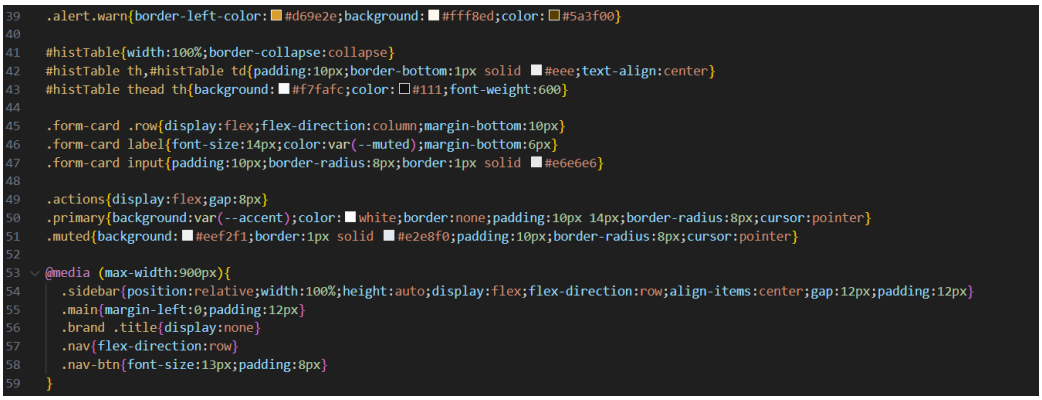
Código de Styles.css



```
:root{
  --bg:#f4f7fa;
  --card:#ffffff;
  --accent:#2f855a;
  --muted:#6b7280;
}

*{box-sizing:border-box}
html,body{height:100%;margin:0;font-family:Inter,system-ui,Segoe UI,Roboto,Arial;background:var(--bg);color:#111}

.layout{display:flex;min-height:100vh}
.sidebar{
  width:240px;
  background:linear-gradient(180deg,#1f6f4e,#165a3f);
  color:#fff;padding:20px;display:flex;flex-direction:column;gap:16px;
  position:fixed;height:100vh;left:0;top:0;
}
.brand{display:flex;align-items:center;gap:12px}
.brand .logo{font-size:24px}
.brand .title{font-weight:700;font-size:18px}
.nav{display:flex;flex-direction:column;gap:8px;margin-top:10px}
.nav-btn{background:transparent;border:none;color:inherit;padding:10px;border-radius:8px;text-align:left;cursor:pointer;font-size:15px}
.nav-btn.active, .nav-btn:hover{background:rgba(255,255,255,0.08)}
.footer{margin-top:auto;font-size:13px;color:rgba(255,255,255,0.85)}

.main{margin-left:260px;padding:28px;flex:1;min-height:100vh}

.section-header{display:flex;align-items:center;justify-content:space-between;margin-bottom:12px}
.section-header h1{margin:0;font-size:20px}
.controls .small{margin-left:8px;padding:8px 10px;border-radius:8px;background:#e6f4ea;border:1px solid #cde8d5;cursor:pointer}

.card{background:var(--card);padding:18px;border-radius:12px;box-shadow:0 8px 20px rgba(16,24,40,0.06);margin-bottom:16px}
.form-card{max-width:560px}

.alerts{display:flex;flex-direction:column;gap:8px;margin-bottom:12px}
.alert{padding:12px;border-radius:8px;border-left:4px solid #e53e3e;background:#fff7f7;color:#802727}
.alert.info{border-left-color:#3182ce;background:#f0f8ff;color:#1b365f}
```

Figura 26.

Código styles.css



```
.alert.warn{border-left-color:#d69e2e;background:#fff8ed;color:#5a3f00}

#histTable{width:100%;border-collapse:collapse}
#histTable th,#histTable td{padding:10px;border-bottom:1px solid #eee;text-align:center}
#histTable thead th{background:#f7fafc;color:#111;font-weight:600}

.form-card .row{display:flex;flex-direction:column;margin-bottom:10px}
.form-card label{font-size:14px;color:var(--muted);margin-bottom:6px}
.form-card input{padding:10px;border-radius:8px;border:1px solid #e6e6e6}

.actions{display:flex;gap:8px}
.primary{background:var(--accent);color:white;border:none;padding:10px 14px;border-radius:8px;cursor:pointer}
.muted{background:#eef2f1;border:1px solid #e2e8f0;padding:10px;border-radius:8px;cursor:pointer}

@media (max-width:900px){
  .sidebar{position:relative;width:100%;height:auto;display:flex;flex-direction:row;align-items:center;gap:12px;padding:12px}
  .main{margin-left:0;padding:12px}
  .brand .title{display:none}
  .nav{flex-direction:row}
  .nav-btn{font-size:13px;padding:8px}
}
```

- **Representación Gráfica en tiempo real (dashboard.js):** La lógica de generación y actualización de gráficos dinámicos fue implementada en dashboard.js, utilizando la librería Chart.js.
    - Los gráficos se actualizan de manera asíncrona mediante consultas periódicas al backend.

- Se representan las variaciones de temperatura y humedad en tiempo real, asegurando un monitoreo continuo.

Figura 27.

Parte del código del dashboard.js

```
// dashboard logic: gráfico, alertas, navegación
let mainChart = null;
let viewMode = "unified"; // "unified" | "temp" | "hum"

const lblsLimit = 20; // mostrar últimos N registros

// Helpers UI
function setApiStatus(text){
  document.getElementById('apiStatus').innerText = text;
}

function showAlert(message, type="error"){
  // type: error | info
  const container = document.getElementById('alerts');
  const div = document.createElement('div');
  div.className = 'alert';
  if (type === 'info') div.classList.add('info');
  if (type === 'warn') div.classList.add('warn');
  div.innerText = message;
  container.prepend(div);
  setTimeout(()=>{ div.remove(); }, 6000);
}

// Render chart given data arrays
function renderMainChart(labels, temps, hums){
  if (mainChart) mainChart.destroy();
  const ctx = document.getElementById('chartMain').getContext('2d');

  const datasets = [];
  if (viewMode === 'unified' || viewMode === 'temp'){
    datasets.push({
      label: 'Temperatura (°C)',
      data: temps,
      borderColor: 'rgb(255,99,132)',
      fill: false,
      tension: 0.3,
      yAxisID: 'y1'
```

- **Configuración de umbrales en la vista principal (thresholds.js):** La interfaz permite al usuario **configurar y actualizar umbrales de control** directamente en la página principal.

  - thresholds.js captura los valores ingresados por el usuario.
  - Envía los datos al backend para su almacenamiento en la base de datos PostgreSQL. Este mecanismo habilita un control operativo inmediato, integrando monitoreo y gestión en una misma vista.

Figura 28.

Código thresholds.js

```
// thresholds form submit handling
document.getElementById('formThresholds').onsubmit = async (e) => {
  e.preventDefault();
  const payload = {
    temp_min: parseFloat(document.getElementById('tempMin').value),
    temp_max: parseFloat(document.getElementById('tempMax').value),
    hum_min: parseFloat(document.getElementById('humMin').value),
    hum_max: parseFloat(document.getElementById('humMax').value)
  };
  try{
    await API.updateThresholds(payload);
    alert('Umbrales actualizados correctamente');
  }catch(err){
    alert('Error actualizando umbrales: ' + err.message);
  }
};

// reset button restores values from server
document.getElementById('btnReset').onclick = async () => {
  const t = await API.getThresholds();
  document.getElementById('tempMin').value = t.temp_min ?? '';
  document.getElementById('tempMax').value = t.temp_max ?? '';
  document.getElementById('humMin').value = t.hum_min ?? '';
  document.getElementById('humMax').value = t.hum_max ?? '';
};
```

- **Gestión avanzada de umbrales (thresholds.html):** Se incluyó una vista complementaria orientada exclusivamente a la configuración de umbrales.
  - Permite recuperar y actualizar los valores almacenados en el servidor.
  - Mejora la usabilidad al ofrecer un espacio dedicado a la gestión de parámetros críticos, sin interferir con los gráficos en tiempo real.

Figura 29.

Código Thresholds.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Umbrales - Sistema de Monitoreo Cultivos Inteligentes</title>
  <link rel="stylesheet" href="assets/css/styles.css" />
  <script src="https://unpkg.com/feather-icons"></script>
</head>
<body>
  <div class="page">
    <header class="page-header">
      <a href="index.html">← Volver</a>
      <h1>Umbrales</h1>
    </header>

    <main class="container">
      <form id="formThresholdsPage" class="card form-card">
        <div class="row">
          <label>Temp. mínima (°C)</label>
          <input type="number" step="0.1" id="tMinPage" />
        </div>
        <div class="row">
          <label>Temp. máxima (°C)</label>
          <input type="number" step="0.1" id="tMaxPage" />
        </div>
        <div class="row">
          <label>Hum. mínima (%)</label>
          <input type="number" step="0.1" id="hMinPage" />
        </div>
        <div class="row">
          <label>Hum. máxima (%)</label>
          <input type="number" step="0.1" id="hMaxPage" />
        </div>

        <div class="row actions">
          <button type="submit" class="primary">Guardar</button>
```

Figura 30.

Código Thresholds.html

```
        </div>
      </form>
    </main>
  </div>

  <script>feather.replace()</script>
  <script src="assets/js/api.js"></script>
  <script>
    // Simple reuse of thresholds.js logic by copy/paste minimal calls
    async function loadPage() {
      const t = await API.getThresholds();
      document.getElementById('tMinPage').value = t.temp_min;
      document.getElementById('tMaxPage').value = t.temp_max;
      document.getElementById('hMinPage').value = t.hum_min;
      document.getElementById('hMaxPage').value = t.hum_max;
    }
    document.querySelector('form').onsubmit = async (e) => {
      e.preventDefault();
      await API.updateThresholds({
        temp_min: parseFloat(document.getElementById('tMinPage').value),
        temp_max: parseFloat(document.getElementById('tMaxPage').value),
        hum_min: parseFloat(document.getElementById('hMinPage').value),
        hum_max: parseFloat(document.getElementById('hMaxPage').value)
      });
      alert('Umbrales actualizados');
    };
    loadPage();
  </script>
</body>
</html>
```

- **Integración de comunicación con el backend (api.js):** El archivo api.js abstrae las llamadas HTTP entre frontend y backend mediante funciones asíncronas basadas en fetch API.
    - Obtiene registros actualizados desde el servidor.
    - Envía configuraciones de umbrales al backend.

Figura 31.

Parte del Código api.js

```
    async getThresholds(){
      const {json} = await tryUrls(endpoints.getThresholds);
      // normalizar nombres comunes
      return {
        temp_min: json.temp_min ?? json.tempMin ?? json.min_temp ?? json.t_min ?? json.umbral_temp ?? json.temp_threshold,
        temp_max: json.temp_max ?? json.tempMax ?? json.max_temp ?? json.t_max ?? json.temp_threshold_max ?? json.temp_threshold,
        hum_min:  json.hum_min  ?? json.humMin  ?? json.min_hum  ?? json.h_min ?? json.umbral_hum ?? json.hum_threshold,
        hum_max:  json.hum_max  ?? json.humMax  ?? json.max_hum  ?? json.h_max ?? json.hum_threshold_max ?? json.hum_threshold
      };
    },

    async updateThresholds(body){
      // probar POST en cada endpoint de update
      for (const u of endpoints.updateThresholds){
        try{
          const res = await fetch(u, {
            method:"POST",
            headers:{"Content-Type":"application/json"},
            body: JSON.stringify(body)
          });
          if (res.ok) return await res.json();
        }catch(e){}
      }
      throw new Error("No se pudo actualizar umbrales en el backend.");
    }
  };
})();
```

Este diseño modular favorece la mantenibilidad y escalabilidad de la aplicación.

la fase de visualización de datos integra de manera armónica la interfaz gráfica, la lógica de interacción y la comunicación con el backend. Su implementación garantiza una experiencia de usuario fluida y confiable, al tiempo que proporciona herramientas estratégicas para la detección temprana de anomalías y la optimización de cultivos **inteligentes** mediante tecnologías IoT.

Fase de Visualización y prueba funcional

En esta etapa se llevaron a cabo las pruebas necesarias para garantizar el correcto funcionamiento del sistema de monitoreo de cultivos automatizados e inteligentes. Inicialmente,

se verificó la conectividad del módulo ESP8266MOD a la red WiFi, así como la transmisión de datos hacia la plataforma en la nube ThingSpeak. Posteriormente, se comprobó la estabilidad de la API desarrollada en Flask y la integridad de los datos almacenados, asegurando que las lecturas de temperatura y humedad se transmitieran de manera confiable y fueran accesibles para su consulta.

Figura 32.

Backend en flask corriendo correctamente

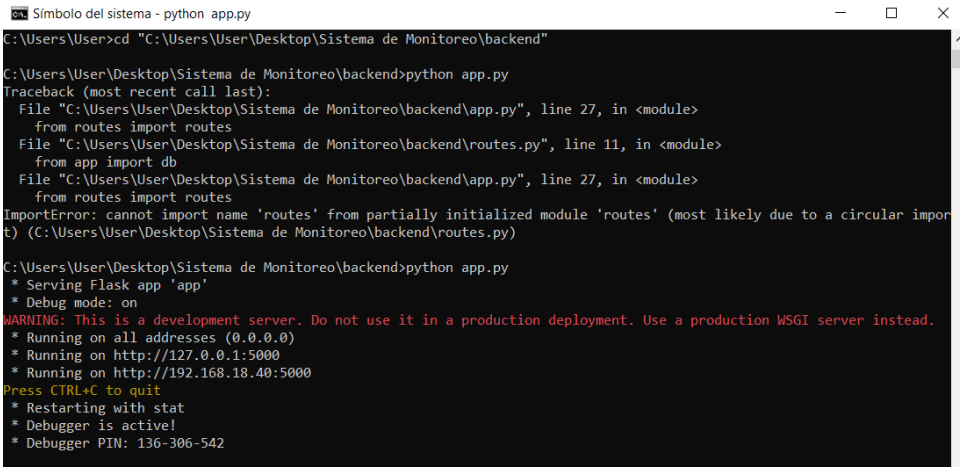
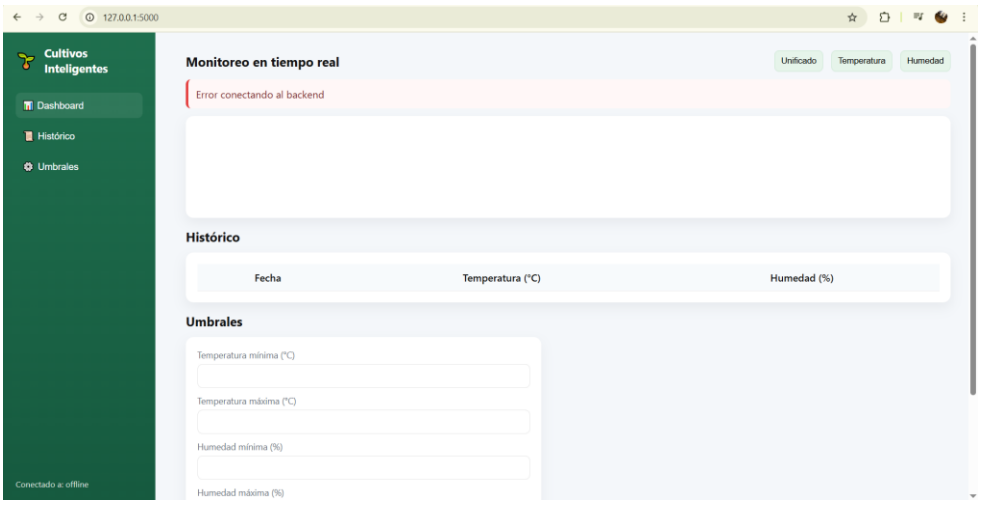


Figura 33.

Primera visualización del Sistema de monitoreo



Durante la primera ejecución del código, tanto del frontend como del backend, se presentó un error en la conexión con el servidor que impedía la visualización de las variables en las gráficas. No obstante, fue posible validar el diseño web y el despliegue correcto de los archivos CSS y HTML. En las siguientes pruebas, al ejecutar el código desde el entorno de desarrollo Spyder con la API en Flask, se logró la conexión en tiempo real con los datos provenientes de los sensores. Los valores fueron enviados al servidor en la nube (ThingSpeak), desde donde posteriormente fueron extraídos por Flask y enviados al frontend para su visualización.

A su vez, el dashboard permitió representar las variables de temperatura y humedad en gráficas dinámicas, con la opción de observarlas de manera unificada o por separado. Una vez validadas las pruebas en la red local, se procedió a desplegar el sistema en la nube con el fin de hacerlo accesible desde cualquier entorno, sin limitar su alcance a la red local.

Figura 34.

Primeros datos de los sensores en la página web.

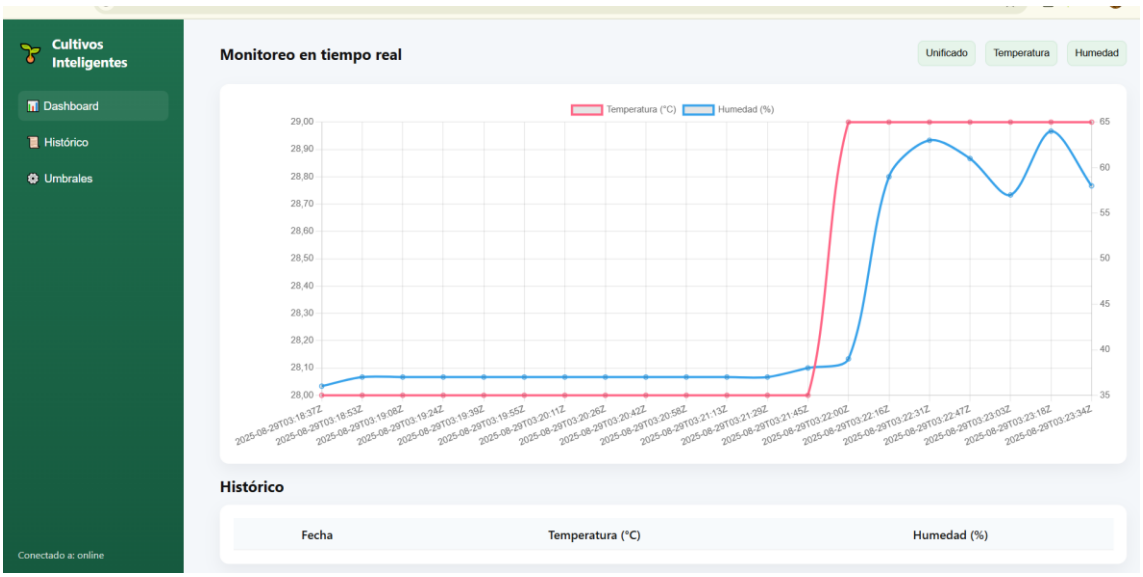


Figura 35.

Gráfica de variables de temperatura y humedad unificadas.

Figura 36.

Grafica de Variables de temperatura y humedad unificada.

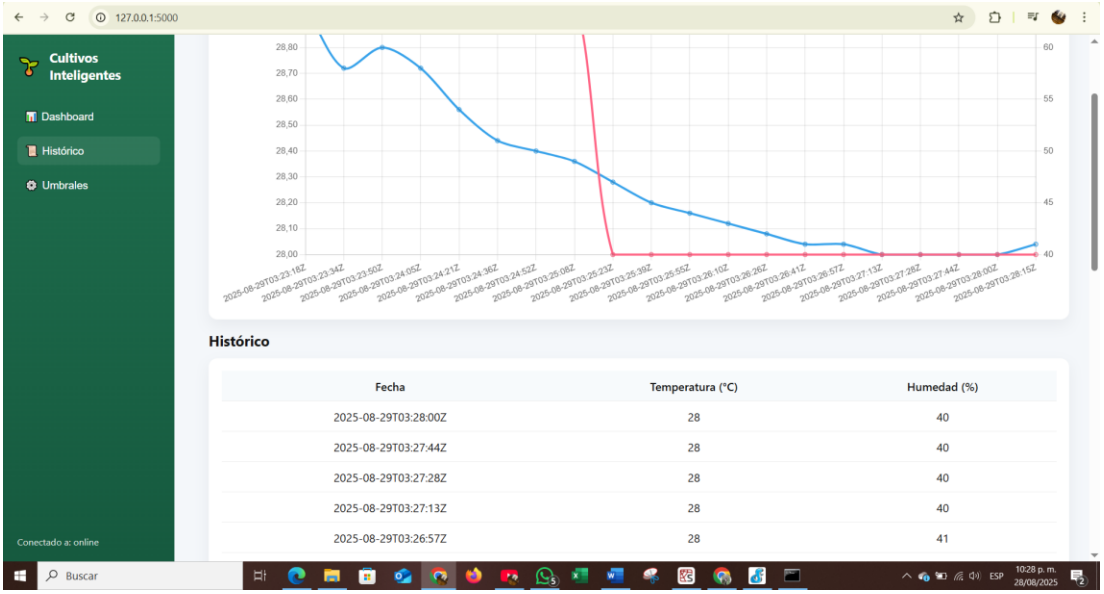


Figura 37.

Gráfica de la variable de temperatura

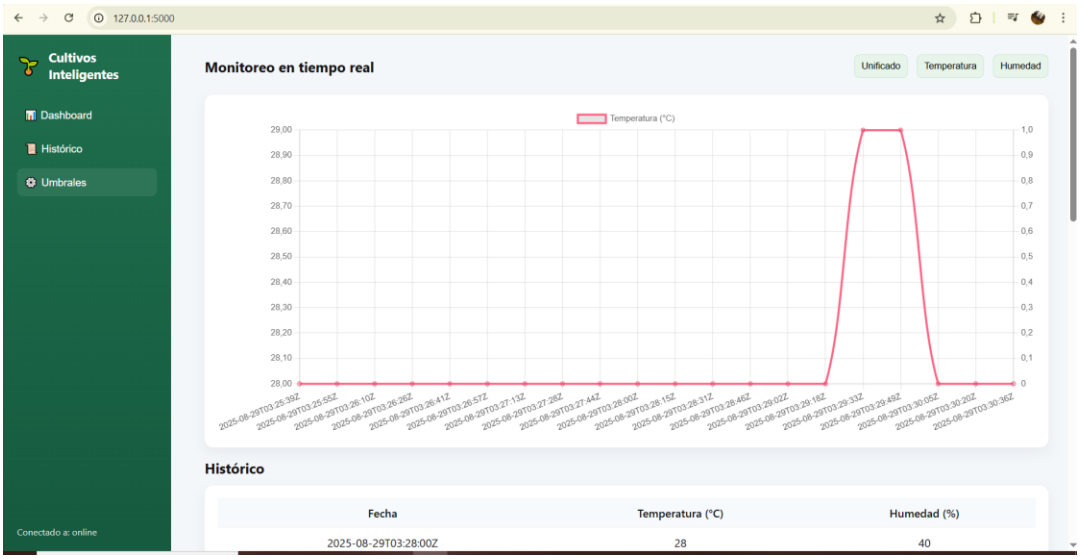


Figura 38.

Gráfica de la variable de humedad

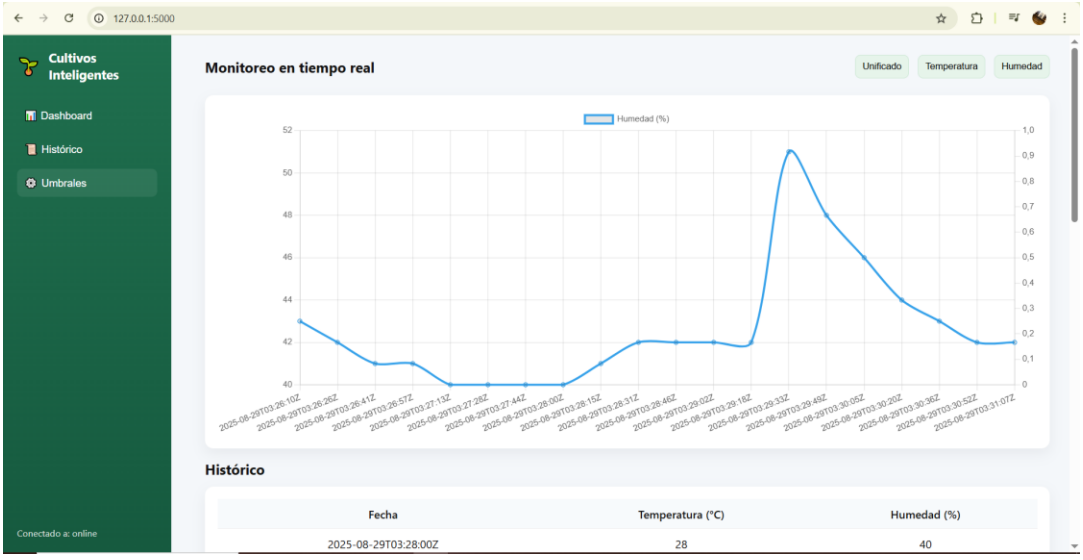


Figura 39.

Visualización del sistema en la red local.

Figura 40.

Datos Históricos del Sistema de Monitoreo visualizado en la red local.

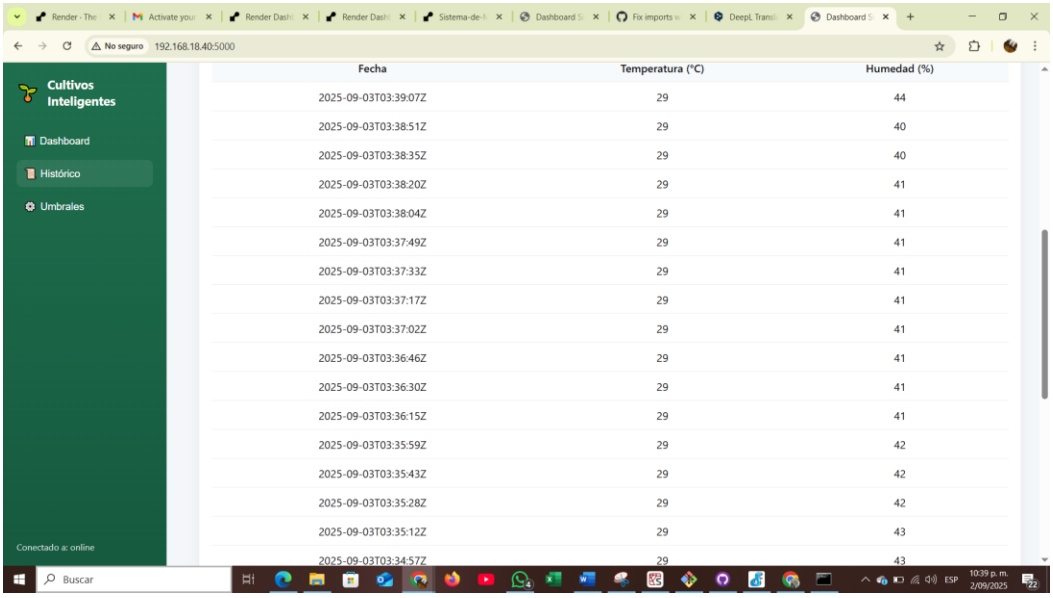
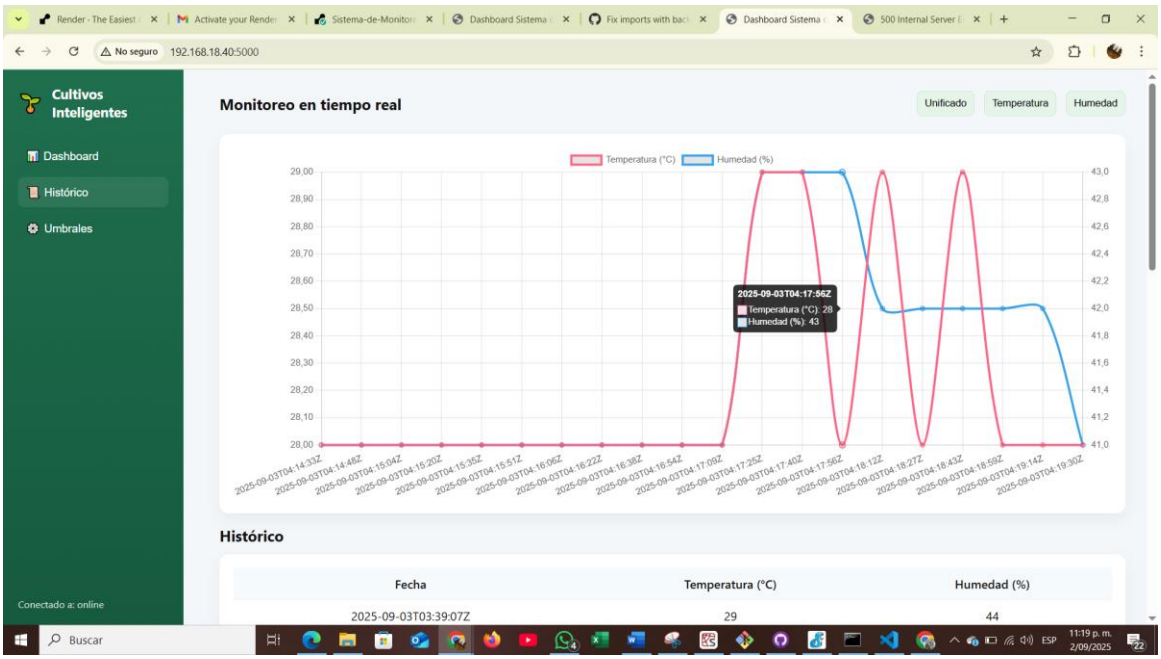


Figura 41.

Comportamiento de las variables de temperatura y humedad.

Figura 42.

Comportamiento de las variables de Temperatura y Humedad en la red local



Ya que se hizo pruebas del código en la red local y se logró la conexión y visualización del sistema, para el despliegue total el código fue gestionado a través de un repositorio en GitHub utilizando Git Bash. Cada actualización del sistema fue subida al repositorio, el cual se integró con la plataforma Render, encargada de alojar y ejecutar la aplicación web sin necesidad de administrar servidores físicos. Render permitió la instalación de las dependencias definidas en el archivo *requirements.txt* y la ejecución del proyecto en la nube mediante Gunicorn, asignando una URL pública y gratuita: https://sistema-de-monitoreo.onrender.com/

Figura 43.

Algunos comandos en Git Bash para realizar Commit

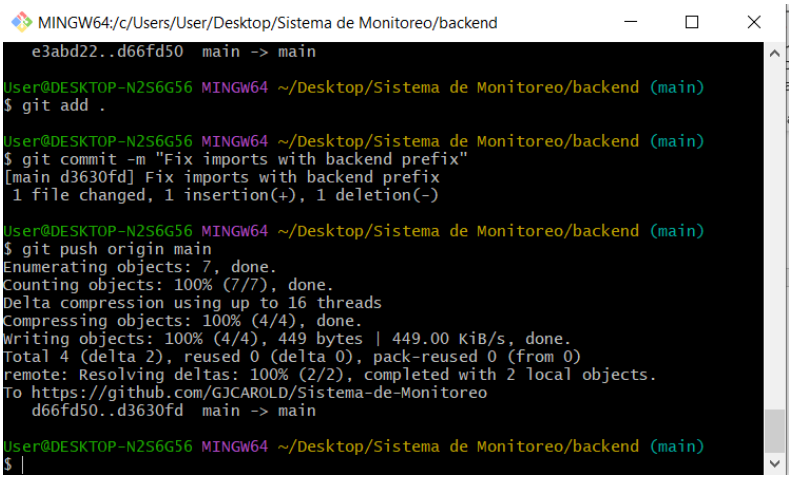


Figura 44.

Repositorio en GitHub

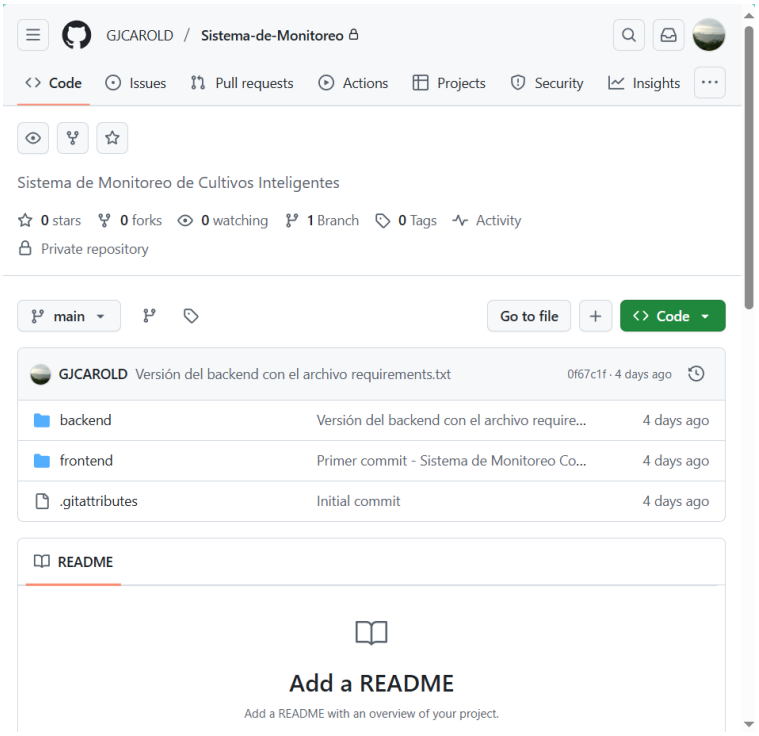


Figura 45.

GitHub Desktop

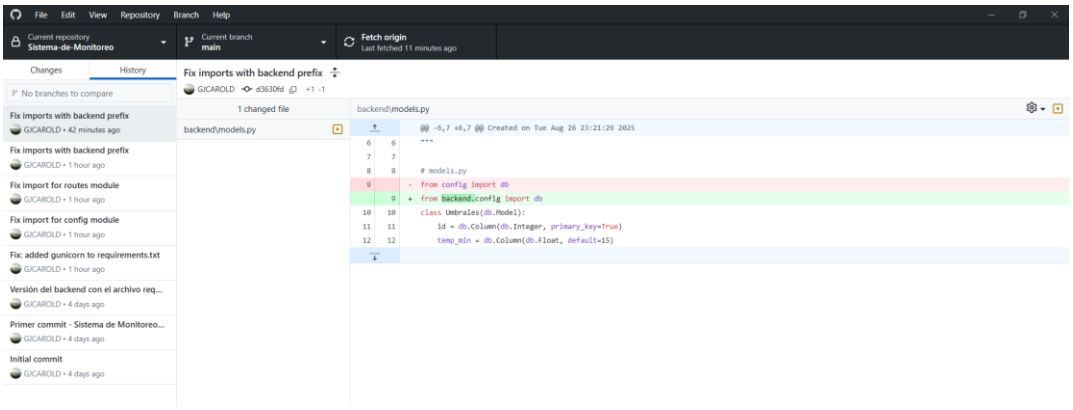


Figura 46.

Commits realizados para ejecutar el sistema de monitoreo en Render

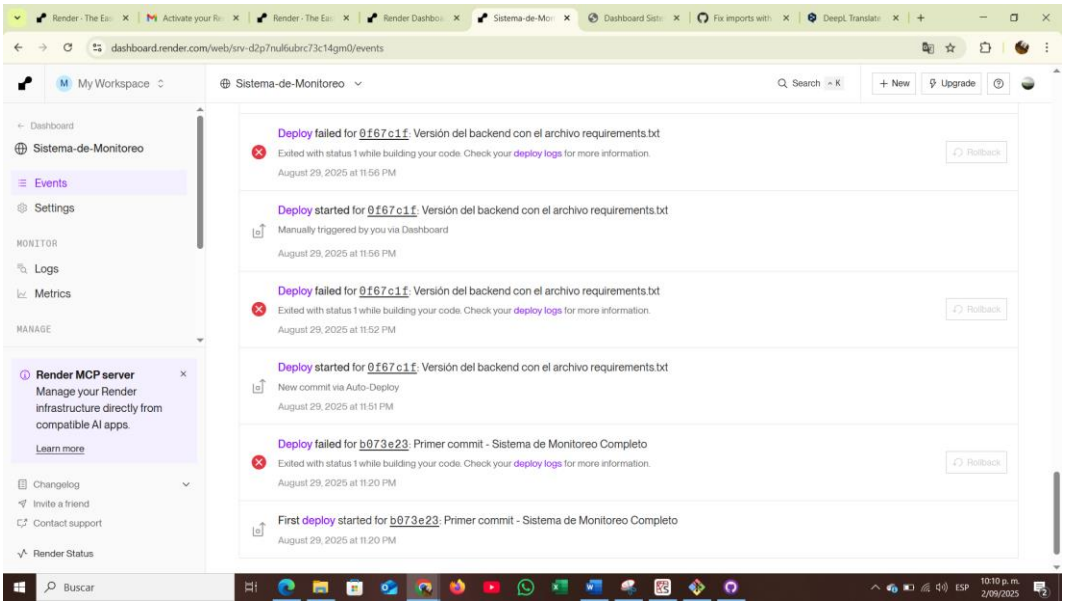


Figura 47.

Commits con error al ejecutar en Render

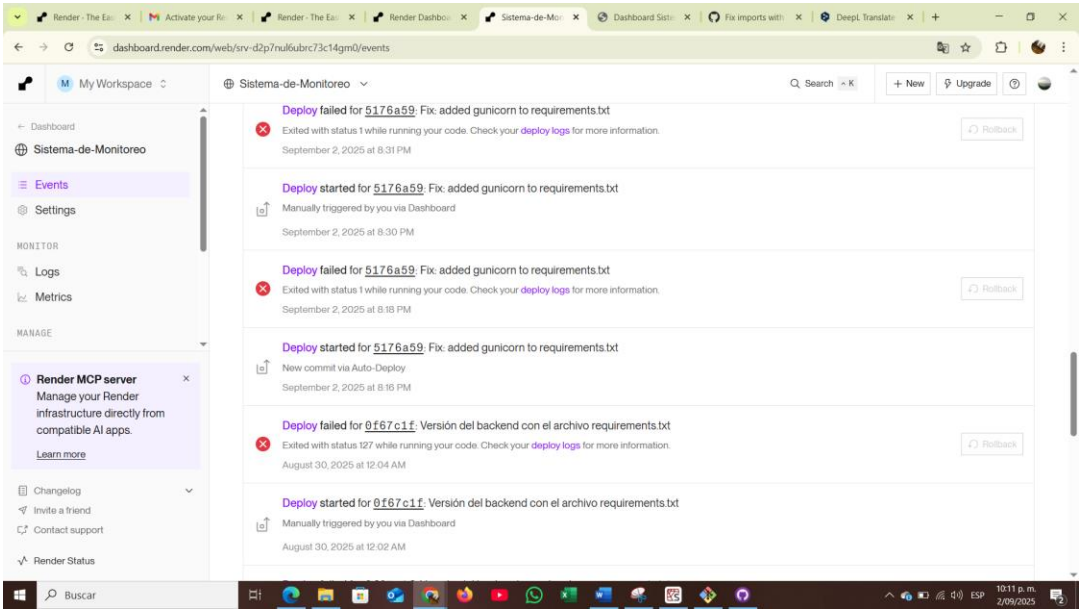


Figura 48.

Código ejecutado correctamente desde Render

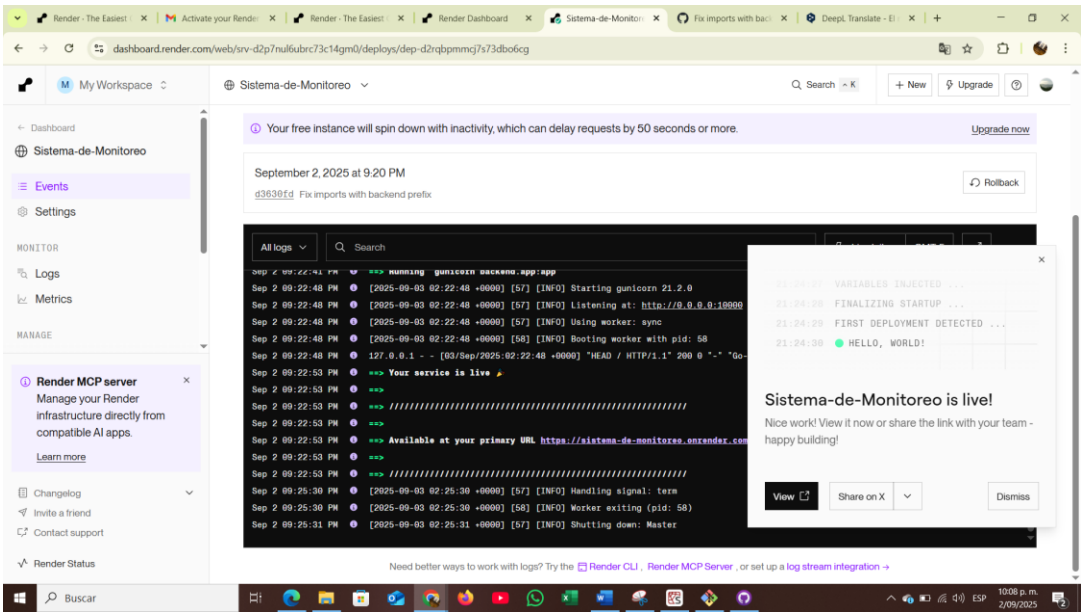
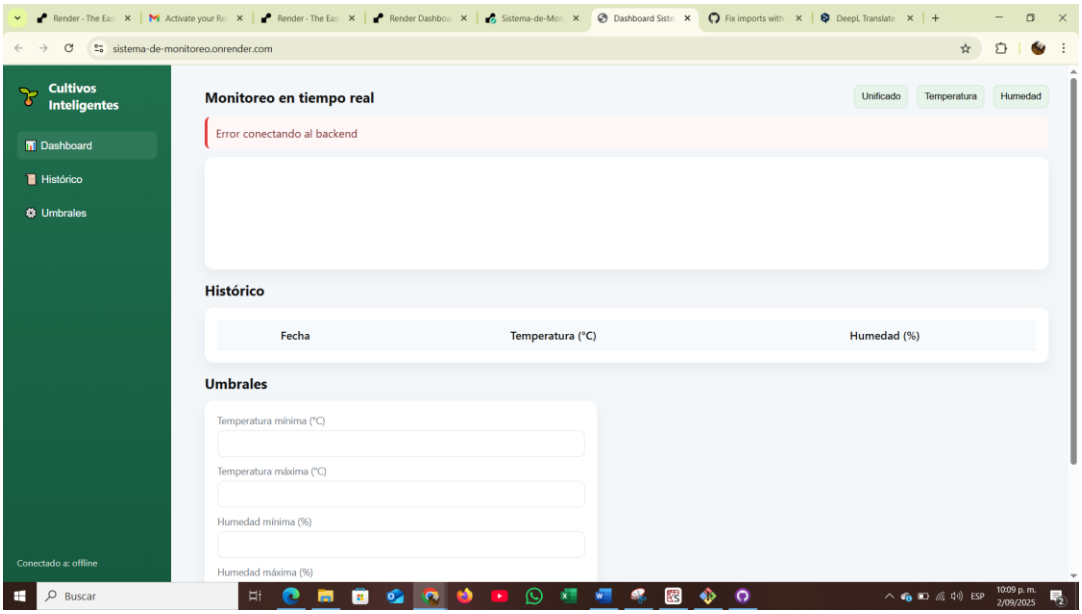


Figura 49.

URL dada en Render para visualizar la Pagina Web del Monitoreo de Cultivos



En el primer intento de despliegue se detectó una falla en la conexión con el backend. Sin embargo, mediante la revisión detallada de los registros (logs) en Render fue posible identificar y corregir el error. Cabe resaltar que el sistema se implementó en el plan gratuito de Render, el cual ofrece soporte para Flask con Python e incluye la posibilidad de utilizar PostgreSQL en lugar de SQLite como base de datos, evitando la pérdida de información tras reinicios del servidor web.

Figura 50.

Configuración de la Base de datos en Render

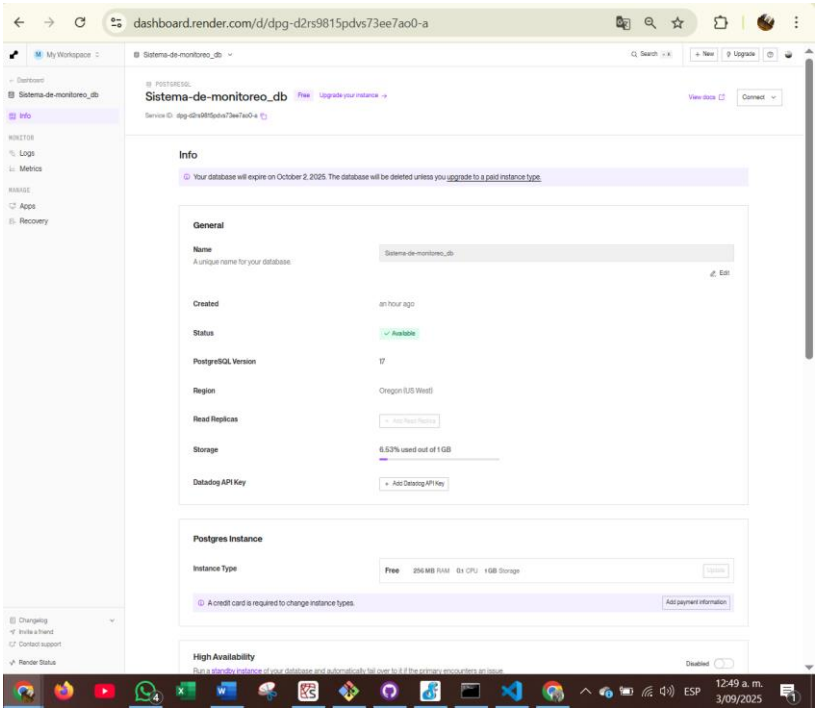


Figura 51.

Visualización de la configuración en Render.

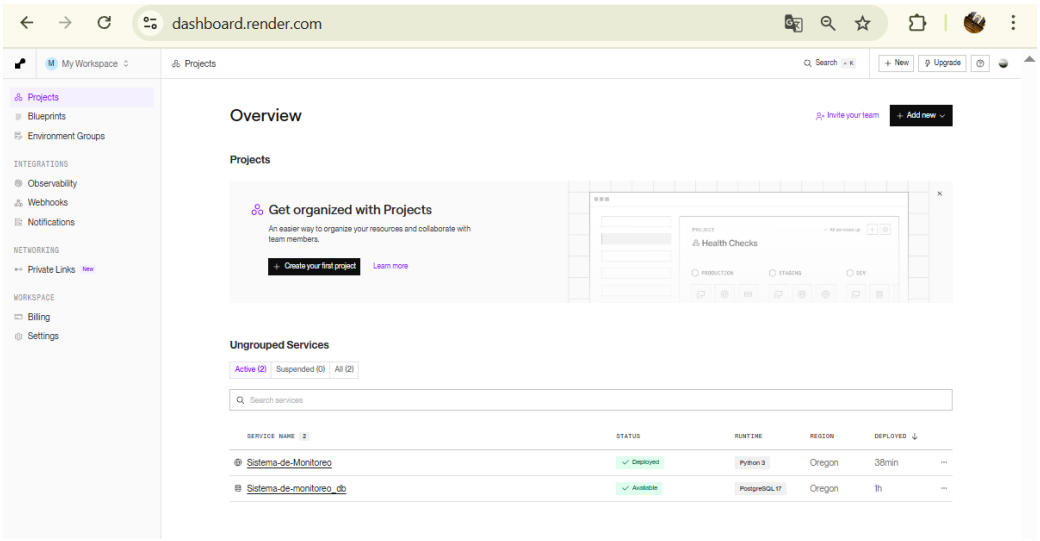


Figura 52.

Visualización de cada evento de despliegue de la página actualizando de acuerdo a como se actualiza el repositorio en GitHub

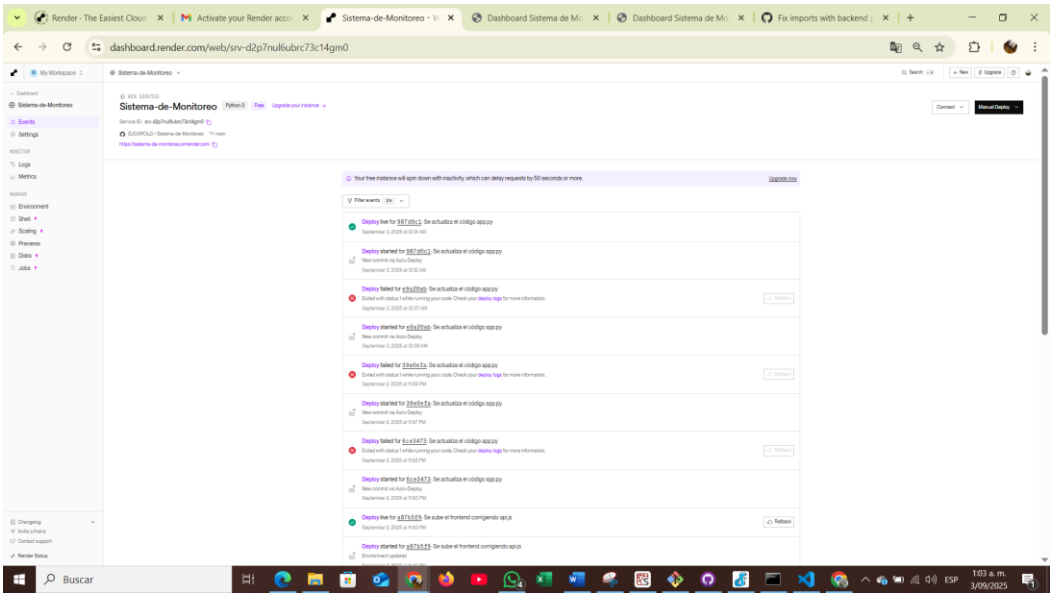


Figura 53.

Despliegue del Sistema de Monitoreo de cultivos Automatizados

Figura 54.

Variable de Humedad

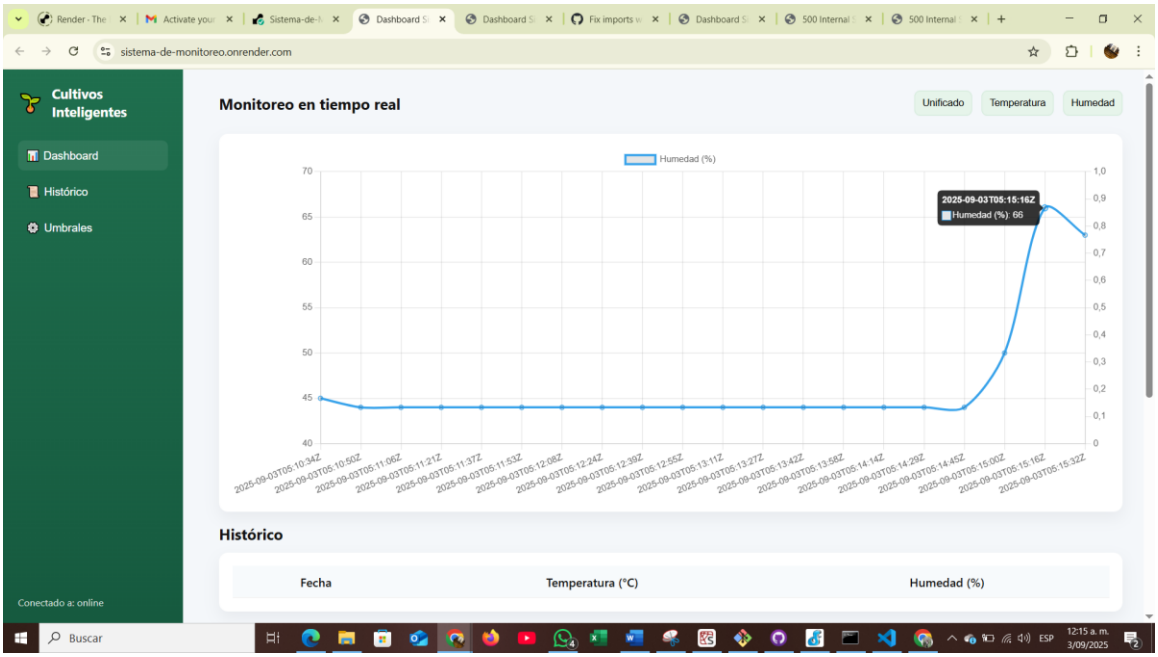


Figura 55.

Variables unificadas

Figura 56.

Visualización de Variables

Figura 57.

Visualización de Variables

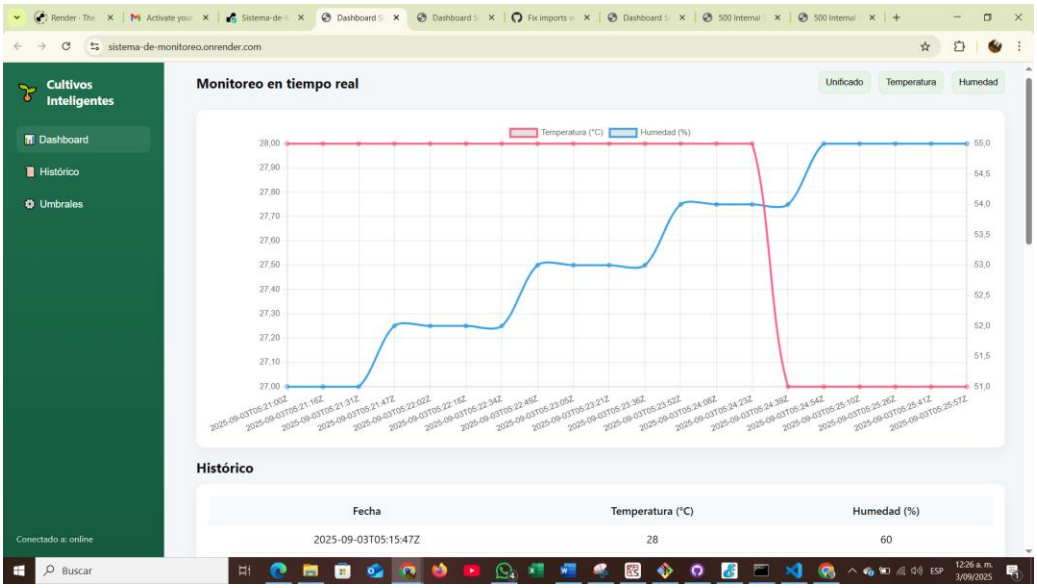


Figura 58.

Recolección de datos históricos de los sensores

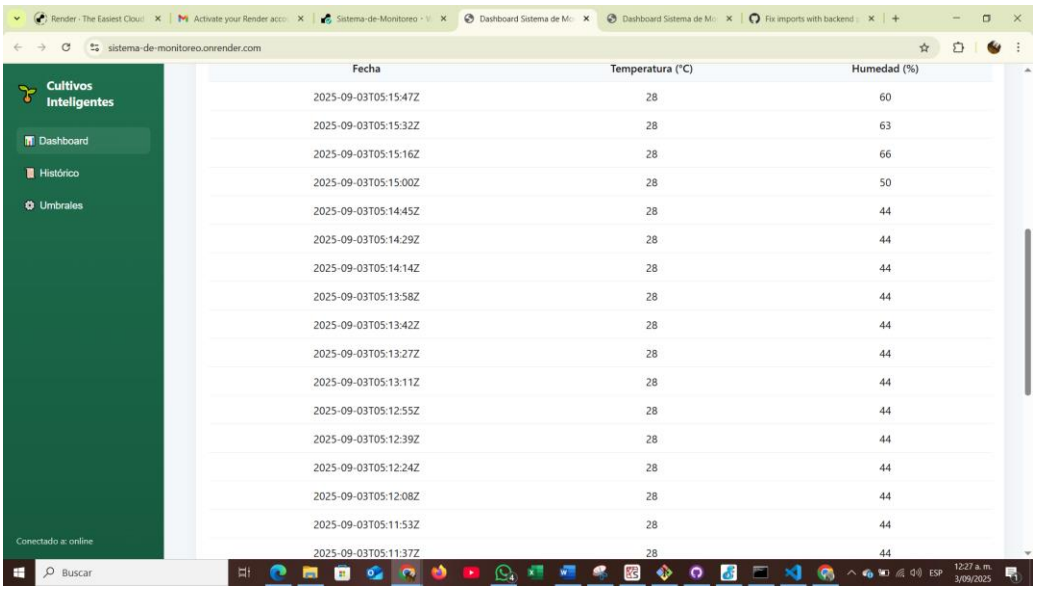


Figura 59.

Despliegue correcto con la ejecución de la base de datos

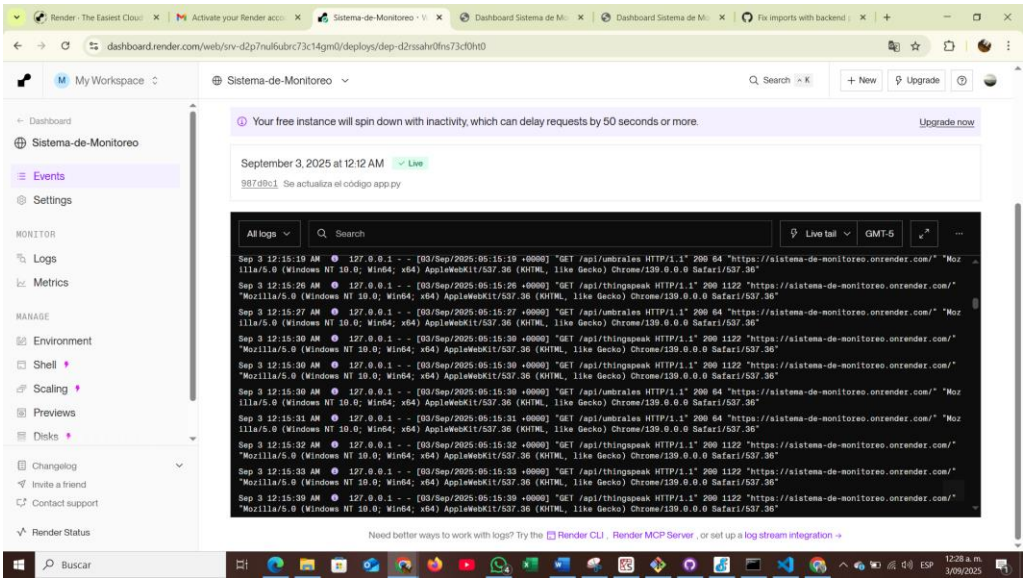


Figura 60.

Configuración de los umbrales de las variables.

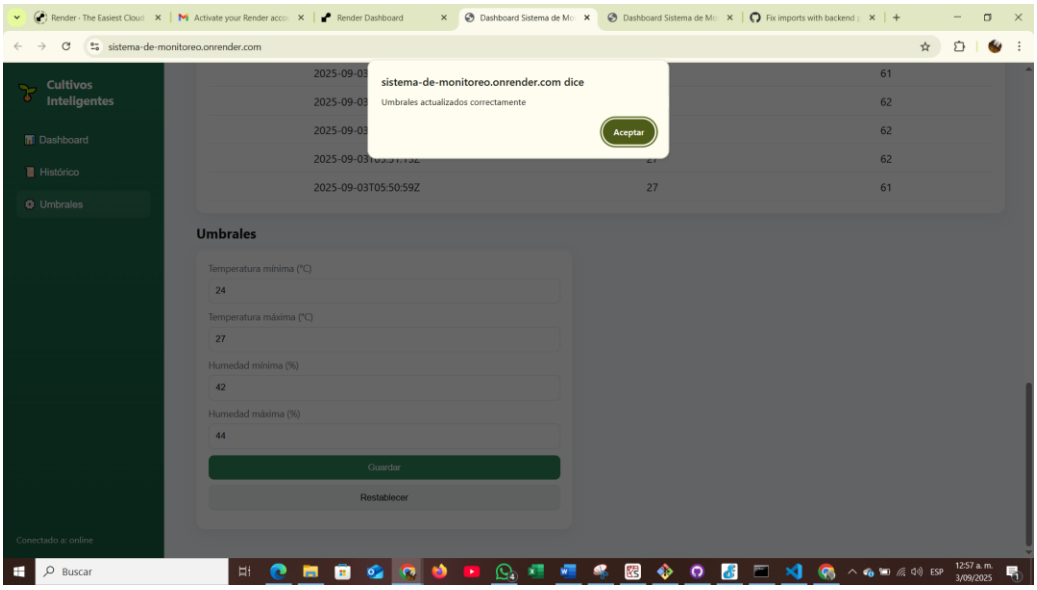


Figura 61.

Alerta de humedad alta.

Figura 62.

Alerta de humedad alta.

Figura 63.

Alerta de humedad baja

Estas pruebas confirmaron el correcto funcionamiento integral del sistema, desde la captura de datos con los sensores hasta la visualización de los mismos en el dashboard, garantizando la disponibilidad de la aplicación en la nube y la accesibilidad desde distintos dispositivos y ubicaciones.

Cronograma

Tabla 11.

Cronograma

Fases	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12
Investigación Y Diseño												
Desarrollo Del Backend												
Desarrollo del Frontend												
Pruebas y Despliegue												

Conclusiones

Se concluye que el proyecto alcanzó exitosamente su objetivo general de desarrollar un sistema de monitoreo web funcional para cultivos inteligentes utilizando Python e IoT. El prototipo fue implementado de manera integral, abarcando desde la recolección de datos de temperatura y humedad con un sensor DHT11 y un microcontrolador ESP8266MOD, hasta la transmisión de esta información a través de la plataforma en la nube ThingSpeak. Finalmente, el sistema fue desplegado con éxito en la nube a través de Render, lo que demuestra la viabilidad técnica de la arquitectura propuesta al permitir la visualización y el monitoreo remoto de las variables del cultivo desde una URL pública.

El desarrollo del proyecto demuestra que Python es una herramienta sumamente versátil y eficaz para crear soluciones de agricultura inteligente de extremo a extremo. En la capa de hardware, se utilizó MicroPython para programar el dispositivo IoT y gestionar la captura de datos. En la capa de procesamiento, el framework Flask fue fundamental para construir un backend robusto con una API REST capaz de recibir, almacenar y procesar la información. Finalmente, en la capa de visualización, librerías como Dash, Plotly y Pandas permitieron crear un dashboard interactivo y en tiempo real, validando la capacidad del ecosistema de Python para integrar hardware, servicios en la nube y análisis de datos en una solución cohesiva.

Este sistema no solo funciona como una solución técnica validada, sino que también sirve como una prueba de concepto tangible sobre cómo las tecnologías de bajo costo y de código abierto pueden contribuir a la modernización del sector agrícola. El prototipo aborda directamente la necesidad de optimizar los procesos de cultivo mediante la toma de decisiones basada en datos, sentando las bases para futuras mejoras como la integración de más sensores, la implementación de algoritmos de inteligencia artificial para análisis predictivo y la

automatización del riego. La arquitectura modular del sistema garantiza su escalabilidad, lo que lo posiciona como un punto de partida sólido para desarrollar herramientas de agricultura de precisión más complejas y accesibles.

Recomendaciones

Se recomienda dar continuidad al proyecto fortaleciendo la seguridad del sistema, tanto en la comunicación entre dispositivos IoT como en la gestión de la base de datos. Es fundamental implementar protocolos de encriptación como TLS/SSL en la transmisión de datos y diseñar políticas de acceso seguro para los usuarios de la plataforma. Esto no solo protegerá la integridad y confidencialidad de la información recopilada, sino que también garantizará la confiabilidad del sistema ante intentos de manipulación externa. En escenarios agrícolas reales, donde la información del cultivo es crítica, la seguridad se convierte en un componente esencial para la sostenibilidad tecnológica.

Asimismo, es aconsejable ampliar el alcance de los sensores integrados, incorporando nuevas variables críticas como luminosidad, conductividad eléctrica, niveles de nutrientes y calidad del agua. Estos parámetros permitirán una supervisión más completa y precisa de las condiciones ambientales que influyen directamente en el rendimiento del cultivo. Además, al integrar modelos predictivos basados en machine learning, se podría anticipar escenarios adversos como sequías, exceso de humedad o deficiencia de nutrientes, brindando alertas tempranas y recomendaciones automáticas para la intervención oportuna. De esta manera, el sistema evolucionaría hacia una herramienta de agricultura de precisión.

Finalmente, se recomienda establecer alianzas estratégicas con productores agrícolas, asociaciones campesinas y centros de investigación especializados. Estas colaboraciones facilitarán la validación del sistema en escenarios de mayor escala y bajo condiciones reales de producción. Al mismo tiempo, permitirán generar retroalimentación valiosa para mejorar la plataforma y ajustarla a las necesidades particulares de cada entorno agrícola. Esta vinculación con actores del sector también favorecerá la transferencia tecnológica, aumentando la

probabilidad de adopción del sistema en comunidades rurales y potenciando su impacto social, económico y ambiental en el marco de la innovación agroindustrial.

Bibliografía

AgroCity. (2022). *Agricultura urbana y aeropónica: Transformando la producción de alimentos en Bogotá.* https://www.agrocity.co

Agrocity. (2022). *Agricultura vertical: la revolución verde de Bogotá. Revista de Innovación y Sostenibilidad Urbana, 4*(1), 72-85. https://www.agrocity.co

Agrosavia. (2019). *La tecnología y la innovación en la agricultura colombiana.* Revista Agrosavia, 1(2), 45-60. https://www.agrosavia.co

Aeroagro. (2021). *La aeroponía como solución para la agricultura urbana en Medellín.* https://www.aeroagro.com.co

AgroTech Campus. (s.f.). *La transformación tecnológica del sector agro*. https://agrotechcampus.com/blog/la-transformacion-tecnologica-del-sector-agro

AgroTIC: Bridging the gap between farmers, agronomists, and merchants through smartphones and machine learning. (2023). https://arxiv.org/abs/2305.12418

AgroTIC: Innovación y Sostenibilidad para el Campo. https://www.mintic.gov.co/portal/inicio/Sala-de-prensa/Noticias/399046%3AInteligencia-Artificial-para-revolucionar-el-sector-agropecuario-el-eje-de-AgroTIC-Innovacion-y-Sostenibilidad-para-el-Campo

Alianza de Bioversity International y CIAT. (s.f.). *Proyectos e Iniciativas Emblemáticas en Colombia*. https://alliancebioversityciat.org/es/proyectos-e-iniciativas-emblematicas-en-colombia

Alotcer. (s.f.). *Comprender MQTT: cómo funciona MQTT en IoT*. https://www.alotceriot.com/es/que-es-mqtt-y-como-funciona-mqtt/

Amazon Web Services. (s.f.). *¿Qué es el MQTT? - Explicación del protocolo MQTT*. https://aws.amazon.com/es/what-is/mqtt/

Arsys. (s.f.). *¿Qué es Visual Studio Code y cuáles son sus ventajas?*. https://www.arsys.es/blog/que-es-visual-studio-code-y-cuales-son-sus-ventajas

Bolaños, J., & Rodríguez, S. (2021). *Blockchain en la cadena de suministro agrícola: el caso del café colombiano. Revista de Comercio Internacional y Tecnología, 14*(3), 56-71. https://doi.org/10.5678/rcit.v14i3.5678

Bootstrap Team. (2024). *Bootstrap Documentation*. https://getbootstrap.com/

Botta, A., De Donato, W., Persico, V., & Pescapé, A. (2016). Integration of Cloud computing and Internet of Things: A survey. Future Generation Computer Systems, 56, 684–700. https://doi.org/10.1016/j.future.2015.09.021

DataCamp. (s.f.). *How to Use SQLite in Python*. https://www.datacamp.com/tutorial/sqlite-in-python

Carvalho, D., Gomes, L., & Rodrigues, J. J. (2017). Towards Smart Farming and Sustainable Agriculture with IoT. In 2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS).

Carvalho, A., Martins, T., & Lima, J. (2017). Detection of plant diseases using computer vision and artificial intelligence. Computers and Electronics in Agriculture, 142, 131-139.

Chanchí-Golondrino, G. E., Ospina-Alarcón, M. A., y Saba, M., (2022). Sistema IoT para el monitoreo de variables climatológicas en cultivos de agricultura urbana. *Revista Científica*, *44*(2), 257-271

Chart.js Contributors. (2024). *Chart.js Documentation*. https://www.chartjs.org/

Cifuentes Vargas, J. (2021). Sistema IoT Para Control y Monitoreo Remoto del Robot FarmBot Uniandes. Universidad de los Andes. Disponible en: http://hdl.handle.net/1992/50657

Corpoica (Agrosavia). (2018). *Implementación de Big Data y la agricultura sostenible en Colombia. Revista de Ciencia y Tecnología Agrícola, 7*(5), 88-102. Recuperado de https://www.agrosavia.co

Dhingra, D., Kumar, N., & Joshi, H. (2019). **A review of applications of machine learning and deep learning in agriculture**. *International Journal of Computer Applications*, *178*(8), 1–7.

DigitalOcean. (2020). *How To Use Flask with SQLite*. https://www.digitalocean.com/community/tutorials/how-to-use-flask-with-sqlite

Dey, R. (s.f.). *Live Server - Visual Studio Marketplace*. https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer

EAFIT. (s.f.). *Tan cerca y tan lejos de la agricultura 4.0 en Colombia*. https://www.eafit.edu.co/investigacion/noticias/Paginas/tan-cerca-y-tan-lejos-de-la-agricultura-4-0-en-colombia.aspx

El País. (2024, 5 de diciembre). *Ricardo Uribe, el 'tropicalizador' del aguacate Hass*. Recuperado de https://elpais.com/america-colombia/branded/los-lideres-de-colombia/2024-12-05/ricardo-uribe-el-tropicalizador-del-aguacate-hass.html

El País. (2024, 5 de diciembre). *Joe Tohme, el libanés que hizo realidad un banco de semillas para combatir el hambre global desde Colombia*. https://elpais.com/america-colombia/branded/los-lideres-de-colombia/2024-12-05/joe-tohme-el-libanes-que-hizo-realidad-un-banco-de-semillas-para-combatir-el-hambre-global-desde-colombia.html

El País. (2024, 5 de diciembre). *Yuly Galindo, la creadora de los 'cultivos por internet' que promueven el comercio justo en el agro*. https://elpais.com/america-colombia/branded/los-lideres-de-colombia/2024-12-05/yuly-galindo-la-creadora-de-los-cultivos-por-internet-que-promueven-el-comercio-justo-en-el-agro.html

Esquivel-Godoy, D. J., Nuño-Maganda, M. A., Hernández-Mier, Y., & Polanco-Martagón, S. (2022). Módulos de supervisión y automatización de un sistema hidropónico mediante lógica difusa y visión por computadora. *Difu100ci@, Revista De difusión científica, ingeniería Y tecnologías*, *16*(3), 15-22. http://difu100cia.uaz.edu.mx/index.php/difuciencia/article/view/277

Edureka. (2019). *What is Python Spyder IDE and How to use it?*. https://medium.com/edureka/spyder-ide-2a91caac4e46

Fazenda Futuro. (2020). *Innovación y sostenibilidad en la agricultura urbana con aeroponía.* https://www.fazendafuturo.com.co

Fontagro. (s.f.). *Proyectos*. https://www.fontagro.org/new/proyectos/

Flask. (s.f.). *Flask-Login Documentation*. https://flask-login.readthedocs.io/

Fedecafé. (2021). *Innovación tecnológica para la producción cafetera.* Boletín de Innovación Agropecuaria, 12(3), 23-28. https://www.fedecafe.com.co

Fundación Ceres. (2020). *Capacitación en técnicas aeropónicas para el desarrollo rural.* https://www.fundacionceres.org

Generation Computer Systems, 108, 273–283. https://doi.org/10.1016/j.future.2020.02.039

Gómez, J. E., Castaño, S., Mercado, T., Fernandez, A., & Garcia, J. (2018). Sistema de internet de las cosas (IoT) para el monitoreo de cultivos protegidos. *Ingeniería E Innovación*, *5*(1). https://doi.org/10.21897/23460466.1101

González, J. A., & Pérez, M. L. (2023). Monitoreo IoT en un Sistema Aeropónico para el cultivo de la Lechuga Orgánica. *Revista de Tecnología Agrícola,* 15(2), 45-58. https://www.researchgate.net/publication/370574426_Monitoreo_IoT_en_un_Sistema_Aeroponico_para_el_cultivo_de_La_Lechuga_Organica

González, M., Toro, L., & Rivas, J. (2023). *Agricultura inteligente con Python en viñedos chilenos*. Revista Latinoamericana de Tecnología Agrícola, 9(1), 56–67.

Grinberg, M. (2018). *Flask Web Development (2nd ed.)*. O'Reilly Media

Gupta, R., Sharma, V., & Sood, M. (2021). Smart irrigation system using Raspberry Pi and Python. International Journal of Advanced Research in Computer and Communication Engineering, 10(4).

InfoWorld. (2013). *Why you should use SQLite*. https://www.infoworld.com/article/2337363/why-you-should-use-sqlite-3.html

Impacto TIC. (2024, 11 de octubre). *Tecnología en el Agro colombiano: Innovación desigual en el campo*. https://impactotic.co/micrositios-tic/impacto-eco/ecnologia-en-el-agro-colombiano-innovacion-desigual-en-el-campo/

Jayashree, S., Rajalakshmi, P., & Usha, R. (2018). *Farm Management Systems: A Review*. Agricultural Reviews, 39(1), 1–7.

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70–90. https://doi.org/10.1016/j.compag.2018.02.016

López, R. A., & Martínez, D. F. (2022). Monitoreo remoto de temperatura, humedad y radiación lumínica en cultivos aeropónicos. *Ingeniería USBMed*, 13(1), 25-38. https://revistas.usb.edu.co/index.php/IngUSBmed/article/view/6334

Madapathage Don, WS, Ahmed, MR, Siraj, M., Anjum, R., Sha, HH, Raja Rani, T. (2024). Monitoreo del crecimiento de las plantas en fábricas de plantas: una solución inteligente de IoT. En: Arefin, MS, Kaiser, MS, Bhuiyan, T., Dey, N., Mahmud, M. (eds) Actas de la 2.ª Conferencia internacional sobre big data, IoT y aprendizaje automático. BIM 2023. Apuntes de la conferencia sobre redes y sistemas, vol. 867. Springer, Singapur. https://doi-org.bibliotecavirtual.unad.edu.co/10.1007/978-981-99-8937-9_65

MathWorks. (2023). *ThingSpeak IoT analytics platform*. MathWorks

Másmela, J. O., Romero-Perdomo, F., & Galvis, C. U. Tecnologías emergentes para el agro y su aplicación en Colombia.

McKinney, W. (2018). *Python for Data Analysis (2nd ed.)*. O'Reilly Media

MicroPython. (2023). *MicroPython Documentation*. https://docs.micropython.org/

Microsoft. (s.f.). *Using Git source control in VS Code*. https://code.visualstudio.com/docs/sourcecontrol/overview

Microsoft. (s.f.). *Get started with web development using Visual Studio Code*. https://learn.microsoft.com/en-us/training/modules/get-started-with-web-development/

Miguelgrinberg.com. (s.f.). *The Flask Mega-Tutorial Part XVI: API Implementation*. https://blog.miguelgrinberg.com/

Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia. (s.f.). *Inteligencia Artificial para revolucionar el sector agropecuario, el eje de*

Ministerio de Minas y Energía de Colombia. (2019). *Uso de energías renovables en la agricultura colombiana.* https://www.minminas.gov.co

Ministerio de Agricultura y Desarrollo Rural de Colombia. (2020). *Tecnologías para la innovación agrícola: Agricultura de precisión en Colombia.* https://www.minagricultura.gov.co

Mozilla Developer Network. (2024). HTML, CSS, and JavaScript documentation. https://developer.mozilla.org/

Netafim. (s.f.). *NetMCU™ control de riego*. https://www.netafim.co/campo-abierto/agricultura-digital/control/netmcu/

Netafim. (s.f.). *Del cultivo al café más rápido con riego de precisión*. https://www.netafimusa.com/es-mx/agriculture/cultivos/cafe/

Netafim. (s.f.). *Soluciones de fertirrigación para cada aplicación*. https://www.netafimcentroamerica.com/productos-y-soluciones-para-riego-de-precision/tools/fertirrigacion/

Netafim. (s.f.). *Desafíos globales y soluciones efectivas: El rol del riego de precisión en la agricultura moderna*. https://www.netafim.co/blog/desafios-globales-y-soluciones-efectivas-el-rol-del-riego-de-precision-en-la-agricultura-moderna/

Netafim. (s.f.). *El riego de precisión será clave en la agricultura sustentable*. https://www.netafim.com.mx/blog/el-riego-de-precision-sera-clave-en-la-agricultura-sustentable/

Netafim. (s.f.). *Riego de precisión: poder para el agricultor*. https://www.netafim.com/es-ec/riego-de-precision/

Netafim. (s.f.). *Soluciones de riego de precisión para huertos*. https://www.netafim.com/es-ec/productos-y-soluciones/sistemas-de-riego-de-presicion-para-huertos

Netafim. (s.f.). *Monitoreo GrowSphere™*. Recuperado de https://www.netafim.com.mx/digital-farming/agricultura-digital/monitoreo/

Netafim. (s.f.). *Productos y soluciones para riego de precisión*. https://www.netafimusa.com/es-cr/agriculture/products-and-solutions/

Paluch, P. (2019). *Choosing the Right Python Framework for Your Web Application*. GeeksforGeeks. https://www.geeksforgeeks.org/choosing-the-right-python-framework-for-your-web-application/

Pandora FMS. (2021). *¿Qué es MQTT? El protocolo más utilizado para IoT*. https://pandorafms.com/es/it-topics/que-es-mqtt/

Pérez, C., & Hernández, M. (2020). *El impacto de la agricultura de precisión en los cultivos de arroz en la región Caribe colombiana. Revista Colombiana de Agricultura, 17*(4), 201-210. https://doi.org/10.1234/rca.v17i4.1234

Plotly. (2023). *Dash Documentation*. https://dash.plotly.com/

Plotly. (2023). *Plotly Python Open Source Graphing Library*. https://plotly.com/python/

Python Software Foundation. (2023). *The Python Standard Library*. https://docs.python.org/

Python Software Foundation. (s.f.). *sqlite3 — DB-API 2.0 interface for SQLite databases*. En *Python Documentation*. Recuperado de https://docs.python.org/3/library/sqlite3.html

Ranveer, K., Ramesh, R., & Patil, A. (2018). Automated irrigation system using IoT and Python. International Journal of Engineering Research and Technology, 7(6).

Ranveer, A., Kumar, S., & Mishra, R. (2018). Smart Irrigation System Using Arduino and Python. International Journal of Engineering Research & Technology (IJERT), 7(9).

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning* (2nd ed.). Packt Publishing.

Real Python. (2021). *How to Scale Flask Applications*. https://realpython.com/scaling-flask/

Rojas, J. P. (2023). IoT e Inteligencia Artificial en la Agricultura. GitHub. https://github.com/juanp-rojash/IoT-e-Inteligencia-Artificial-en-la-Agricultura

Romero, A., & Sánchez, J. (2021). *La implementación de sistemas aeropónicos en el Valle del Cauca: Un modelo de agricultura sostenible. Revista de Innovación Agrícola,* 18(3), 56-72. Recuperado de https://www.valledelcauca.gov.co

SG Buzz. (2010). *SQLite: La Base de Datos Embebida*. En *Software Guru*. https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida

SEIDOR. (s.f.). *Tecnología en el Campo: Transformando la Agricultura Colombiana*. https://www.seidor.com/es-co/blog/tecnologia-en-el-campo-transformando-la-agricultura-colombiana-con-seidor

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. IEEE Internet of Things Journal, 3(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

SQLite. (s.f.). *About SQLite*. https://www.sqlite.org/about.html

Spyder Development Team. (s.f.). *Spyder: The Scientific Python Development Environment*. https://www.spyder-ide.org/

Spyder Development Team. (s.f.). *Scientific Computing and Visualization with Spyder*. https://docs.spyder-ide.org/current/workshops/scientific-computing.html

Spyder Development Team. (s.f.). *Spyder: The Scientific Python Development Environment*. https://www.spyder-ide.org/

Stack Overflow. (2017). *Using Plotly in Python Spyder IDE*. https://stackoverflow.com/questions/45714150/using-plotly-in-python-spyder-ide

Torres Cortes, G. (2021). Diseño e implementación de un sistema de adquisición y registro de señales e imágenes con tecnología IoT para el seguimiento de las condiciones de cultivos hidropónicos de lechuga.

Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M.-J. (2017). Big Data in Smart Farming – A review. Agricultural Systems, 153, 69–80. https://doi.org/10.1016/j.agsy.2017.01.023

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.

Vertical Harvest. (2024). Creating the future we want From The Food We Need https://verticalharvestfarms.com/

Universidad Nacional de Colombia. (2021). *Investigación en sistemas aeropónicos para el cultivo urbano.* https://www.unal.edu.co

Universidad Piloto de Colombia. (2022). Prototipo de monitoreo y control para cultivos de fresa basado en IoT. Repositorio Unipiloto.

Universidad Industrial de Santander. (2023). Plataforma IoT para monitoreo de cultivos de tomate cherry en invernadero. Noesis UIS.

Universidad El Bosque. (2022). Cultivo vertical automatizado con interfaz de monitoreo en tiempo real. Repositorio Institucional.

Universidad Distrital Francisco José de Caldas. (2022). Prototipo de monitoreo agrícola con tecnología LoRa e IoT. 1Library.co.

Universidad Cooperativa de Colombia. (2022). Plataforma IoT para monitoreo de cultivos hidropónicos en Montería. Repositorio UCC.

Zapata, J. R. (s.f.). *Visualización de Datos con Python*. https://joserzapata.github.io/courses/python-ciencia-datos/visualizacion/

Zhang, Y., Deng, R. H., & Weng, J. (2020). Towards Secure and Efficient Data Acquisition for Edge-IoT-Cloud Collaborative Agriculture Systems. Future

Zhang, Y., Wang, Y., & Sun, Y. (2020). Design and implementation of a smart agricultural system using open-source platforms. Computers and Electronics in Agriculture, 175, 105580.

Zhang, X., Liu, Y., Wang, W., & Wang, L. (2021). **Plant disease recognition via deep learning: A review**. *Plant Methods, 17,* Article 72. https://doi.org/10.1186/s13007-021-00762-1

Zhou, Y., Zhang, J., Chen, Q., & Cao, Y. (2017). **Prediction of irrigation demand using Scikit-learn machine learning models**. *Procedia Engineering, 154*, 1260–1265. https://doi.org/10.1016/j.proeng.2016.07.544