Solución para el aprendizaje, implementación y seguimiento de cultivos urbanos medicinales

Director(es)

Dianalin Neme Prada

Ivan Rodrigo Romero Florez

David Mauricio Jiménez López & Wilson Sneider Contreras

Mayo 2021

Universidad Antonio Nariño

Facultad de Ingeniería.

Especialización en Ingeniería de Software

Índice

Índice de Figuras

Índice de Tablas

Introducción

La pandemia generada por el COVID-19 ha demostrado a nivel mundial y más aún en Colombia la falta de capacidad de atención del sistema de salud. Para optimizar dicha capacidad es posible hacer uso de la medicina tradicional y complementaria por parte de los usuarios, la cual fue reconocida por la OMS en el congreso sobre Medicina Tradicional celebrado en Beijing en 2008 y en por el Ministerio de Salud a través del documento "Lineamientos técnicos para la articulación de las medicinas y las terapias alternativas y complementarias, en el marco del sistema general de seguridad social en salud". Sin embargo, es claro que en el sector urbano no se cuenta con una cultura agrícola que facilite el cultivo de plantas medicinales.

Por lo anterior, se propone en el presente proyecto la construcción de un prototipo de sistema de información aplicando conceptos de ingeniería de software como la arquitectura, la calidad y la seguridad, que facilite a los usuarios llevar a buen término cultivos de plantas medicinales a través de contenidos pedagógicos al respecto y un sistema de medición que permita hacer seguimiento de dichos cultivos.

Para la realización del sistema de información se realiza un análisis de productos similares ya existentes en el mercado y la academia con el fin de definir el valor agregado de la solución a desarrollar y así precisar la arquitectura más adecuada para el producto de software, posteriormente se realizan los diseños de software correspondientes que permiten dar inicio a la construcción, la cual, es evaluada con los escenarios de calidad y las pruebas de seguridad más relevantes del sistema con el objetivo de crear un producto mínimo viable aplicando una metodología de desarrollo ágil acorde con las con la naturaleza del proyecto y sus componentes , las condiciones de asilamiento actuales, los perfiles y conocimientos previos del equipo de trabajo, apoyados en el uso de herramientas online para la gestión del proyecto.

Formulación del problema

La Organización mundial (OMS) de la salud en las declaraciones de Alma-Ata y Beijing de 2008, que fueron adoptadas en el Congreso de la OMS acerca de Medicina Tradicional, se considera tal práctica como uno de los ejes importantes a considerar en la atención básica en salud, y con mayor relevancia países en vía de desarrollo como Colombia. Adicionalmente en la Declaración de Beijing instan a la comunidad internacional a través de los gobiernos, organizaciones de salud y demás actores a velar por que la medicina tradicional y que esta se utilice adecuadamente como un componente importante que contribuye a la salud de todas las personas.

En Bogotá, D.C, existe un comercio amplio de plantas medicinales a través de las plazas de mercados, soportado por un conocimiento etnobotánico que sobre medicina tradicional ha trascendido de generación en generación y que vale la pena dar valor más aún con el acceso al sistema de salud es difícil, pues, si bien Colombia posee una cobertura en salud del 95%, de acuerdo con el Banco Interamericano de desarrollo, solo el 30% de los asegurados pueden realmente de forma oportuna acceder a la atención primaria del sistema. El poco oportuno acceso al sistema de sumado a restricciones de movilidad ya sea por el estado de salud del paciente o por decisiones estatales como las actuales cuarentenas dadas por la emergencia sanitaria a raíz del Covid-19 las plantas medicinales cultivadas en espacios urbanos se convierten en una alternativa muy viable para el tratamiento de patologías leves que puedan ser manejadas en casa.

Ahora bien, en entornos urbanos por dinámicas y condiciones de ciudad que existen tales como actividades económicas, costumbres y ambientales entre otros, el conocimiento sobre cultivos no es igualmente transmitido y relevante como en entornos rurales. Además, las

condiciones de cultivo difieren en estos dos entornos dadas las diferentes condiciones ambientales. Por otra parte, no existe en las ciudades esa tradición agrícola necesaria que permita de manera empírica conocer las necesidades de una planta y empoderar a los ciudadanos inexpertos en la implementación de cultivos urbanos a través de conceptos, técnica y/o métodos de cultivo. Si bien áreas de conocimiento como la botánica permiten también llevar cultivos urbanos a buen término, estos conocimientos no se dan de manera expedita o no despiertan el mismo interés a las personas dada su naturaleza científica.

Así las cosas, se revela la necesidad de crear un medio por el cual un ciudadano cualquiera logre llevar a buen término su cultivo urbano de plantas medicinales para el manejo de algunas patologías en casa sin necesidad de trasladarse. Por esta razón consideramos que el medio más apropiado, dada su masificación, disponibilidad, inmediatez de la información y posibilidades didácticas y de consulta es una aplicación móvil que le permita al usuario:

1. Aprender a implementar un cultivo urbano de determinada planta medicinal de su interés a través de material didáctico digital
2. Realizar un seguimiento del estado del cultivo que se determinará a través de un hardware que permite medir las condiciones en tiempo real de las plantas para realizar las acciones de mejora respectivas.

Objetivo General

Construir un sistema de información que permita al sector urbano el aprendizaje y monitoreo del cultivo de huertos medicinales, con el fin de facilitar el seguimiento continuo de la evolución de dichos cultivos.

Objetivos Específicos

Construir un prototipo electrónico funcional de procesamiento sensórico, para capturar niveles de humedad, pH y temperatura de una planta.

Desarrollar un prototipo funcional de aplicación móvil en Android que permita al usuario acceso a contenidos pedagógicos para la creación de cultivos urbanos y el monitoreo de estos.

Definir e implementar la arquitectura de software que permita soportar la adquisición, almacenamiento, procesamiento y disposición de los datos generados por el sistema de sensórica.

Especificar y evaluar los escenarios de calidad relacionados con la disponibilidad del sistema de sensorica, mantenibilidad y tiempo de respuesta de consultas de seguimiento de cultivos.

Realizar un análisis de riesgos que permita aplicar un proceso de desarrollo seguro.

Marco de referencia

5.1. Estado del arte

Dada la naturaleza del proyecto, se plantea el análisis de alternativas en 3 áreas de acción que en alguna medida comparten el mismo objetivo del proyecto y manejan componentes similares tanto de software como de hardware, las áreas mencionadas son aplicaciones móviles, proyectos académicos y soluciones comerciales.

Aplicaciones móviles. En el mercado existen muchas aplicaciones cuya promesa es facilitar al usuario el éxito de un cultivo urbano, ya sea vertical, horizontal, hidropónico, tales como Maceto Huerto, Ihuerting Start , Hidroponic Tech que si bien llevan al usuario a la consecución de su huerta con un buen contenido didáctico, funciona como un tutorial interactivo que no maneja datos directamente sobre el cultivo, lo cual presenta un sesgo en los contenidos en la medida que como cualquier ser vivo, presenta particularidades que se pueden salir del programa.

Proyectos Académicos. Existen algunas soluciones de carácter académico que buscan igualmente automatizar cultivos para un escenario particular, cuentan con los sensores, métricas y automatizaciones necesarias para alcanzar el objetivo, llevar a buen término el cultivo. Si bien son muy rigurosos en el componente electrónico en cuanto a la captura de datos analógicos como luz, temperatura, humedad y su correspondiente digitalización llevan el uso del software móvil a la parametrización del sistema y captura de datos en una red o un servidor locales, como es el caso de Agrodroyd donde indican el uso de XAMPP para un acceso localhost.

De forma similar está el proyecto Prototipo De Control Para Un Cultivo De Tomate Cherry En Un Invernadero, tesis de grado de la U Católica de Colombia plantean la

comunicación a través de una LAN entre el dispositivo móvil y el sistema sensorial. Este tipo de implementaciones plantean una limitante en términos de despliegue que no permitirían llegar a un target más grande de usuarios como se podría hacer a través de una aplicación publicada

Soluciones comerciales. En este aspecto existen varias soluciones de alto costo dada la precisión y complejidad del hardware utilizado y están más enfocados a la producción del cultivo urbano sin generar conocimiento didáctico en el usuario sobre la implementación de un cultivo urbano lo cual plantea una alta dependencia del dispositivo, algunos de ellos con sus ventajas, desventajas y diferencia con el proyecto a desarrollar son:

Farmbot, que puede apreciar en la figura 1, es un huerto robótico diseñado por estudiantes de ingeniería mecánica de la universidad politécnica estatal de california, cuya función es cultivar y llevar a cabo el seguimiento de estos cultivos para su germinación. El sistema funciona a partir de coordenadas cartesianas, para huertos cuyo tamaño se puede dimensionar bajo especificaciones de su distribuidor. Este sistema funciona a partir de rieles y deslizadores que inyectan las semillas teniendo en cuenta parámetros como la profundidad y la distancia entre semillas, que varían dependiendo del tipo de cultivo. Controlados de manera electrónica por microprocesadores como Arduino y Rasperry Pi, además de tener también componentes sensóricos como “sensores de suelo”.

El software que posee permite copiarse, modificarse y distribuirse libremente. Su uso funcional consta de una distribución en donde, de manera interactiva, el usuario selecciona el tipo de semilla y la posición en el huerto para que, posteriormente, el sistema de inyección de semillas trabaje. Adicionalmente ofrece información básica sobre cuidados que requieran la atención del usuario y que el Farmbot no sea capaz de llevar a cabo.

El dispositivo cuenta con la misma necesidad de generar una comunicación en tiempo real con los cultivos, sin embargo, su objetivo principal es que sea capaz de funcionar en condiciones precarias, por lo que su procesamiento y censado debe ser también precario, para optimizar el uso de energía. Este dispositivo no cuenta con una aplicación o manual de usuario capaz de capacitar al usuario final, puesto que se tiene como premisa que dicho usuario sólo va a consumir el alimento que se le proporcione.

Nuestro proyecto llevará a cabo una capacitación continua al usuario para que este pueda contar con las habilidades de cultivos básicas. Es decir, el mantenimiento de los cultivos será llevado a cabo por el usuario y no por un tercero que monitoree el dispositivo.

Figura 1

Cultivo urbano implementado por FarmBot

Nota. Fuente: FarmBot Inc. (2020).

Marko es un sistema electrónico de cultivo en disposición vertical para la cocina diseñado en España cuyo diseño se puede apreciar en la figura 2, su función es llevar a cabo el seguimiento de cultivos de manera automatizada, sin necesidad de la actuación del usuario. Este sistema suministra todas las condiciones necesarias para la germinación de todo tipo de plantas, ya sea también cultivos que se encuentren fuera de temporada.

Su sistema consta de una disposición vertical de módulos llamados "Módulos planta", los cuales pueden ser desacoplados de la base en caso de que se requiera reemplazar el cultivo o llevar a cabo otro tipo de tarea manual. Cada módulo cuenta con su sistema sensorico e informa en tiempo real sobre las propiedades de los cultivos. La base de Marko, provee de luz y agua a las plantas dependiendo de sus necesidades, además, como es bien sabido que algunas plantas requieren diferentes intensidades o frecuencias lumínicas, esta propiedad se puede manejar desde un aplicativo móvil, el usuario selecciona el tipo de planta y el dispositivo aplica los parámetros lumínicos requeridos.

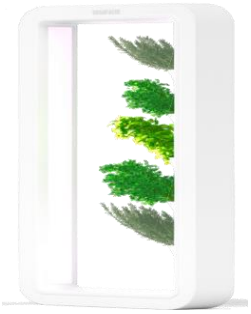
Este es el dispositivo más completo que encontramos en el mercado con respecto a la automatización de cultivos urbanos. Sin embargo, se observa que a pesar de tener en cuenta la necesidad de llevar a cabo un seguimiento continuo con los cultivos, no se hace un seguimiento al usuario. Es decir, no se le dan las técnicas y recomendaciones necesarias para que este adquiera los conocimientos necesarios. Por ende, si se presenta algún problema en el cultivo que el sistema no sea capaz de comprender por alguna falla de cualquier tipo, el usuario no va a saber cómo reaccionar.

El sistema por desarrollar en este proyecto no sólo llevará a cabo un seguimiento en tiempo real a la planta, sino que también capacitará al usuario sobre técnicas y recomendaciones que sus cultivos requieran de manera continua, no sólo se le presentará un manual fijo al usuario,

sino que, dependiendo de las necesidades del cultivo o de inconvenientes que se vayan desarrollando en este, se le asesore al usuario sobre qué puede hacer él directamente para solucionarlo.

Figura 2.

Huerto Vertical Makro, Copyright (2019)



Nota. Fuente: Makro Ink (2019)

TekRarium: es un sistema auto sostenible de producción de alimento y agua enfocado para condiciones climáticas extremas. Es un proyecto diseñado y desarrollado en el taller vertical "Prometeo" que incluye estudiantes y profesionales de la universidad Jorge Tadeo y la universidad Central de Bogotá. Este dispositivo consta de 4 subsistemas principales los cuales son:

1. Producción de energía eléctrica: A partir de 10 sistemas de paneles solares, se genera la energía eléctrica suficiente para que todo el prototipo funcione correcta y continuamente.

2. Producción de agua potable: Gracias a un sistema de condensación de agua mecánico, se provee de agua potable a los diferentes cultivos y al usuario final.

3. Producción de alimento: El dispositivo cuenta con disposiciones para cultivos consumibles muy básicos de mantener y para granjas de insectos comestibles. De este subsistema

se resalta el módulo electrónico, el cual censa condiciones de estos cultivos o granjas, y las acondiciona con agua o nutrientes para mantenerlos en los estándares de germinación adecuados.

4. Monitoreo remoto: El módulo electrónico, gracias a la adquisición de datos de censado, procederá a enviarlos de manera remota a una base de datos para su análisis, en caso de que se requiera intervención de terceros.

El dispositivo cuenta con la misma necesidad de generar una comunicación en tiempo real con los cultivos, sin embargo, su objetivo principal es que sea capaz de funcionar en condiciones precarias, por lo que su procesamiento y censado debe ser también precario, para optimizar el uso de energía. Este dispositivo no cuenta con una aplicación o manual de usuario capaz de capacitar al usuario final, puesto que se tiene como premisa que dicho usuario sólo va a consumir el alimento que se le proporcione.

Nuestro sistema llevará a cabo una capacitación continua al usuario para que este pueda contar con las habilidades de cultivos básicas. Es decir, el mantenimiento de los cultivos será llevado a cabo por el usuario y no por un tercero que monitoree el dispositivo.

Figura 3

Proyecto TekRarioum

Nota. Fuente: U. Tadeo Lozano (2018)

Terrakit es un masetero automatizado diseñado por una empresa emergente de Málaga, España, cuya función es llevar un seguimiento continuo a un cultivo en específico a partir de sensores implícitos en el equipo y que no requieren ningún tipo de manipulación por parte del usuario. Su diseño es ideal para espacios reducidos y su nicho de mercado está enfocado en usuarios que no tienen tiempo para hacer el seguimiento ni conocimientos sobre la implementación y seguimiento de cultivos.

Además, cuenta con una aplicación móvil que va ligada al masetero que informa al usuario en tiempo real de todas las condiciones del cultivo, lo cual genera una comunicación directa con dicho cultivo y facilita el plan de acción.

Este masetero adquiere la información de las propiedades y necesidades de los cultivos y las transmite para ser interpretadas por el usuario, sin embargo, no se le da una tutoría detallada sobre los requerimientos actuales de la planta y el dispositivo no actúa tampoco en consecuencia.

Nuestro sistema buscará la forma de comunicarse constantemente con el usuario en su propio lenguaje, es decir que le comunicará las necesidades y requerimientos del cultivo de forma concisa y clara, para que se entienda qué hay que hacer, y cómo puede mejorarse o evitarse para futuros cultivos.

Figura 4

Diseño Terrakit

Nota. Fuente: Terrakit. (2018)

5.2. Impacto

Impacto en la sociedad. Se busca tener un impacto, de manera indirecta pero concisa, en todo el sector urbano, partiendo por diferentes puntos, el primero, lograr generalizar conceptos y estándares propios de la agricultura. Es bien sabido que este sector no cuenta con los espacios para llevar a cabo una práctica adecuada como se lograría en el sector rural. Sin embargo, la apropiación de dichas prácticas en el hogar podría generar el acople que se ha perdido con el tiempo, hacia el sector primario y la agricultura.

El segundo punto es, concientizar a los residentes del sector a llevar a cabo buenas prácticas en el ámbito ecológico. Si bien sabemos, una de las problemáticas más notorias en este sector, es la contaminación. Y más allá de lo que intenta aportar nuestro dispositivo, llevar a cabo cultivos en el hogar no sólo generará un impacto en el medio ambiente, sino que le aportará al usuario un sentido de pertenencia hacia su entorno, haciendo que lleve a cabo estas buenas prácticas que contribuyan a su cuidado.

5.3. Componente de innovación

Si bien el tema de la agricultura en el sector urbano ha sido un tema tratado previamente en el desarrollo de productos, aplicaciones o dispositivos electrónicos. Tenemos en cuenta algunos componentes en nuestro proyecto, que resaltan sobre los demás.

Implementar tecnologías de automatización en procesos del día a día es una realidad que avanza de manera exponencial conforme pasa el tiempo. Y las tecnologías que proveen ayudas en procesos de la agricultura rural o urbana, son incontables.

Muchas aplicaciones o productos nos brindan una capacitación sobre conceptos y técnicas para la implementación y el cuidado de cultivos, sin embargo, dada la naturaleza de estos modelos, no pueden llevar a cabo un seguimiento continuo ya que desconocen el proceso

por el que el usuario pasa en tiempo real. Nuestro dispositivo, a partir de la adquisición de datos sensóricos, será capaz de llevar a cabo este seguimiento, informando sobre técnicas y corrección de errores en tiempo real, para lograr una comunicación más directa y especializada.

5.4. Marco Teórico

Plantas Medicinales: Se define como planta medicinal como cualquier especie vegetal que contiene sustancias potencialmente empleadas para fines terapéuticos o cuyos principios activos pueden ser usados como precursores para la síntesis de nuevos fármacos de acuerdo con la OMS (1979).

A nivel global las personas han incrementado el uso de las medicinas y terapias alternativas para el cuidado de la salud, lo cual ha llevado a las diferentes instituciones de salud a nivel mundial a poner en consideración la incorporación en los sistemas sanitarios. Integrar este tipo de medicina a los sistemas de salud ha permitido a las personas tener mayor posibilidad de beneficiarse de atenciones alopáticas o convencionales, tradicionales, complementarias y alternativas, sin entrar en conflicto. La falta de inclusión de las medicinas tradicionales en el Sistema General de Seguridad Social en Salud supone barreras culturales y económicas para acceder a la atención en salud, razón por la cual y por tanto dificulta el goce pleno al derecho a la salud.

Por lo anterior el Ministerio de salud ha definido directrices para el uso de medicina tradicional en el documento “lineamientos técnicos para la articulación de las medicinas y las terapias alternativas y complementarias, en el marco del sistema general de seguridad social en salud” donde se reconoce la importancia del uso de la medicina tradicional

Si bien estos lineamientos aumentan la capacidad del sistema de salud para el manejo de patologías más leves, la oportunidad de acceso al mismo no es el mejor dada la complejidad del

sistema de salud y el déficit de profesionales, esta situación conlleva a la automedicación de medicina moderna, que no es recomendable, y el consumo de preparaciones basadas en plantas tradicionales.

En cuanto la Arquitectura IoT, el Internet de las cosas (IoT) está cambiando la manera en que las personas y las organizaciones se conectan con clientes, proveedores, socios y otras personas. IoT consiste en conectar sensores, actuadores y dispositivos a una red y permitir la recopilación, el intercambio y el análisis de la información generada.

Las innovaciones de hardware, como la Raspberry Pi o Arduino, hacen que sea más viable desarrollar nuevos dispositivos. Los estándares de red para redes de baja potencia, como LoRaWAN o NB-IOT, crean nuevas oportunidades para conectar dispositivos muy pequeños a una red. Se están desarrollando nuevos estándares específicamente para casos de uso de IoT, como MQTT para mensajería, OMA Lightweight M2M para administración de dispositivos o W3C Web of Things y oneM2M para interoperabilidad de servicios. Y, por último, las mejoras significativas en el almacenamiento de datos, el análisis de datos y el procesamiento de eventos permiten respaldar la cantidad de datos generados en implementaciones de IoT a gran escala.

En una solución de IoT típica se identifican 3 segmentos fundamentales:

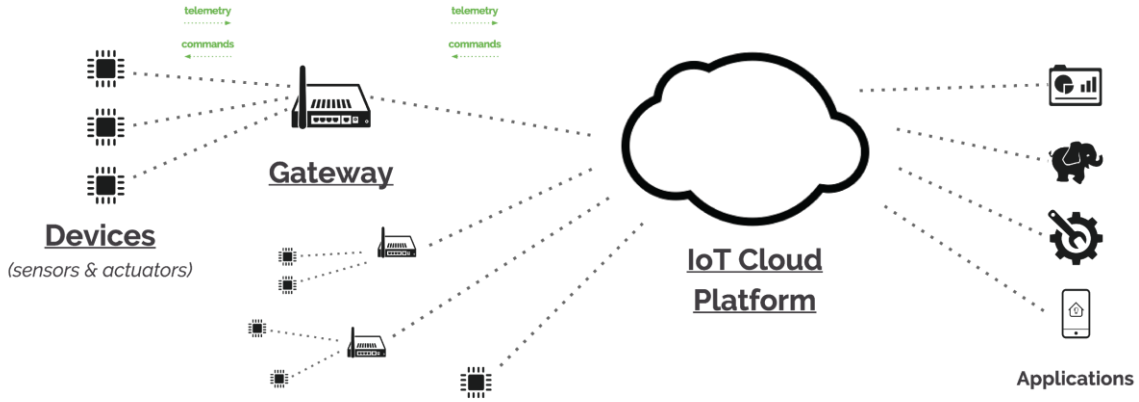
1. Dispositivos
2. Gateway para comunicarse los dispositivos a través de una red
3. Nube IoT, un servidor que ejecuta una plataforma de IoT que ayuda a integrar la información de IoT.

Adicionalmente, en esta arquitectura y la visión humana de IoT, que se pueden apreciar en las figuras 5 y 6, se definen roles para los dispositivos, las puertas de enlace y la plataforma

en la nube, cada uno de estos aprovisiona las características y funcionalidades específicas necesarias para cualquier solución robusta de IoT.

Figura 5

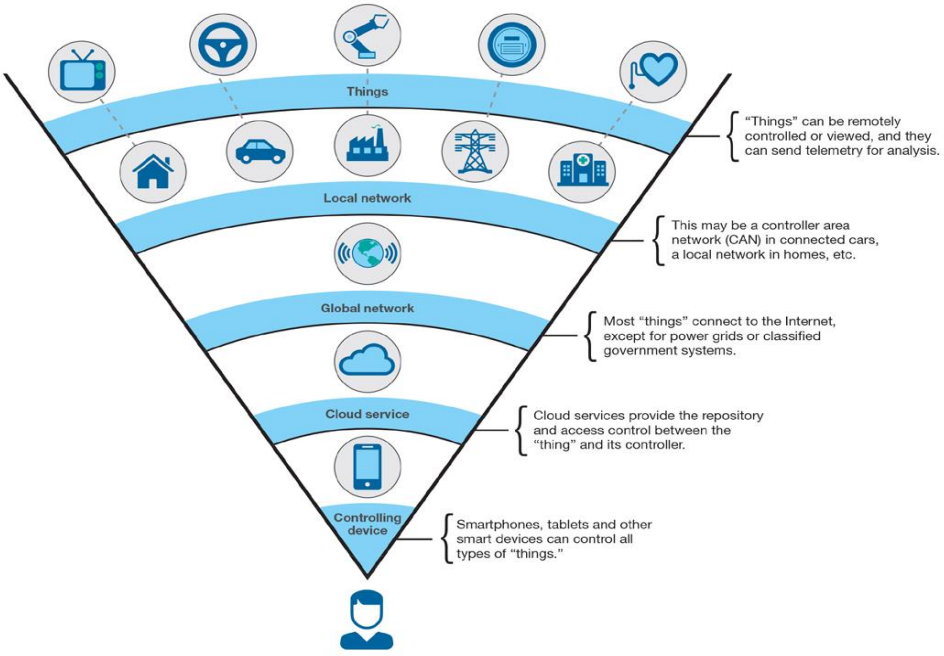
Solución de IoT estándar.



Nota. Fuente: Eclipse Fundation (2019).

Figura 6

Visión humana de Internet de las Cosas.

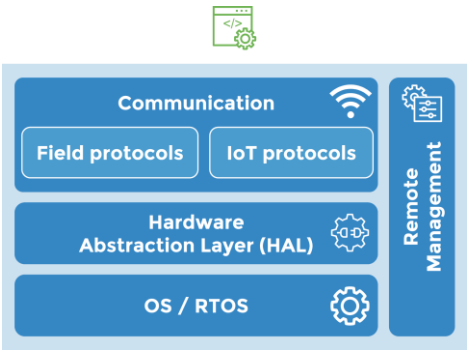


Nota. Fuente: Eclipse Fundation. 2019.

Los dispositivos son el punto de partida para una solución de IoT. Usualmente obtiene los datos e interactúa con el mundo físico. Los dispositivos suelen estar muy limitadas en términos de tamaño o suministro de energía; por lo tanto, a menudo se programan utilizando microcontroladores (MCU) que tienen capacidades muy limitadas. Los microcontroladores que alimentan los dispositivos de IoT están especializados para una tarea específica y están diseñados para producción en masa y bajo costo.

Figura 7

Stack de software en dispositivo IoT.



Fuente. Eclipse Fundation (2019).

El software que se ejecuta en dispositivos basados en MCU tiene como objetivo admitir tareas específicas. Las características clave de la pila de software que se ejecuta en un dispositivo pueden incluir

1. Sistema operativo de IoT: muchos dispositivos se ejecutarán con un servidor 'bare metal', pero algunos tendrán sistemas operativos integrados o en tiempo real que son particularmente adecuados para dispositivos pequeños restringidos y que pueden proporcionar capacidades específicas de IoT.

2. Abstracción de hardware: una capa de software que permite el acceso a las funciones de hardware de la MCU, como memoria flash, GPIO, interfaces seriales, etc.

3. Soporte de comunicación: controladores y protocolos que permiten conectar el dispositivo a un protocolo cableado o inalámbrico como Bluetooth, Z-Wave, Thread, bus CAN, MQTT, CoAP, etc., y permiten la comunicación del dispositivo.

4. Administración remota: la capacidad de controlar de forma remota el dispositivo para actualizar su firmware o monitorear el nivel de la batería.

En cuanto al Gateway, es quien se encarga de punto de enlace un grupo de sensores y/o actuadores para coordinar la conectividad de estos dispositivos entre sí y con una red externa. Puede ser una pieza de hardware como una Raspberry o una funcionalidad de un dispositivo más grande conectado a la red. Por ejemplo, una máquina industrial podría actuar como Gateway.

Por lo general ofrecerá procesamiento de datos "en el borde" y capacidades de almacenamiento manejar la latencia y confiabilidad de la red. Para la conectividad de dispositivo a dispositivo, un Gateway de IoT se ocupa de los problemas de interoperabilidad entre dispositivos incompatibles. Una arquitectura de IoT típica tendría muchos Gateway para administrar una gran cantidad de dispositivos.

Los gateways de IoT dependen cada vez más del software para implementar la funcionalidad principal. Las características clave de una pila de software de puerta de enlace, que pueden apreciarse en la figura 8 incluyen:

1. Sistema operativo: generalmente, un sistema operativo de propósito general como Linux.

2. Contenedor de aplicaciones o entorno de ejecución: por lo general tendrán la capacidad de ejecutar código de aplicación y permitir que las aplicaciones se actualicen dinámicamente. Por ejemplo, una puerta de enlace puede ser compatible con Java, Python o Node.js.

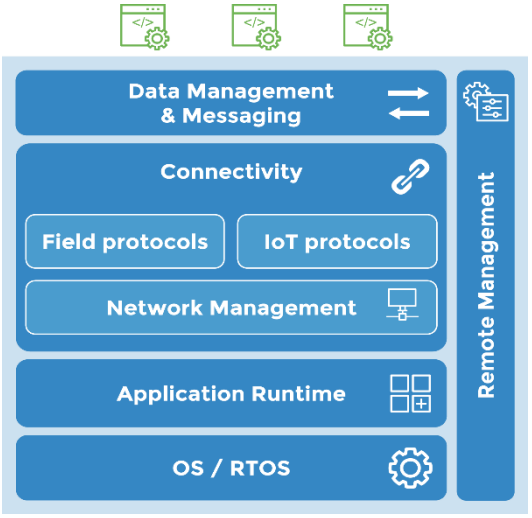
3. Comunicación y conectividad: Deben admitir diferentes protocolos de conectividad para conectarse con diferentes dispositivos (por ejemplo, Bluetooth, Wi-Fi, Z-Wave, ZigBee, Thread). También deben conectarse a diferentes tipos de redes (por ejemplo, Ethernet, celular, Wi-Fi, satélite, etc.…) y garantizar los estándares de calidad de la información.

4. Gestión de datos y mensajería: persistencia local para admitir la latencia de la red, el modo fuera de línea y el análisis en tiempo real en el borde, así como la capacidad de reenviar datos del dispositivo de manera coherente a una plataforma de IoT.

5. Gestión remota: la capacidad de aprovisionar, configurar, iniciar / apagar gateways de forma remota, así como las aplicaciones que se ejecutan.

Figura 8

Stack de software de gateways



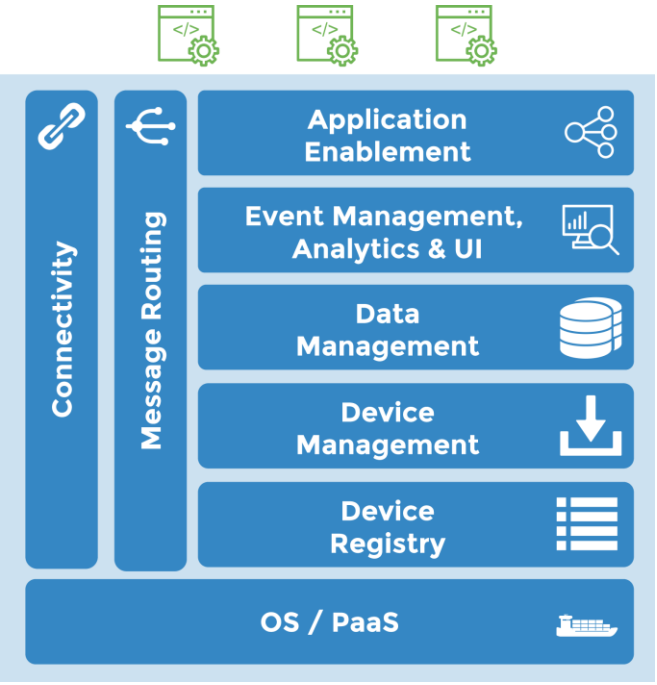
Nota. Fuente: Eclipse Foundation (2019)

Plataforma Cloud IoT. El tercer componente clave es la plataforma cloud, representa la infraestructura de software y los servicios necesarios para habilitar una solución de IoT. Generalmente opera en una infraestructura en la nube (por ejemplo, AWS, Microsoft Azure, etc.) o dentro de un centro de datos empresarial y se espera que se escale tanto horizontalmente, para admitir la gran cantidad de dispositivos conectados como verticalmente a abordar la variedad de

soluciones de IoT. Facilita también interoperabilidad de la solución de IoT con las aplicaciones empresariales existentes y otras soluciones de IoT.

Figura 9

Stack de software Cloud



Nota. Fuente: Eclipse Foundation (2019)

Las características principales de una plataforma cloud para IoT incluyen:

1. Conectividad y enrutamiento de mensajes: las plataformas de IoT deben poder interactuar con una gran cantidad de dispositivos y puertas de enlace utilizando diferentes protocolos y formatos de datos, pero luego normalizarlos para permitir una fácil integración en el resto de la empresa.

2. Administración de dispositivos y registro de dispositivos: un registro central para identificar los dispositivos / puertas de enlace que se ejecutan en una solución de IoT y la capacidad de proporcionar nuevas actualizaciones de software y administrar los dispositivos.

3. Gestión y almacenamiento de datos: un almacén de datos escalable que admite el volumen y la variedad de datos de IoT.

4. Gestión de eventos, análisis e interfaz de usuario: capacidades de procesamiento de eventos escalables, capacidad para consolidar y analizar datos y crear informes, gráficos y paneles.

5. Habilitación de aplicaciones: capacidad para crear informes, gráficos, paneles de control y utilizar API para la integración de aplicaciones.

Funcionalidad de stack o pila cruzada. En los diferentes stack de una solución de IoT hay una serie de características que deben tenerse en cuenta para cualquier arquitectura de IoT, que incluyen:

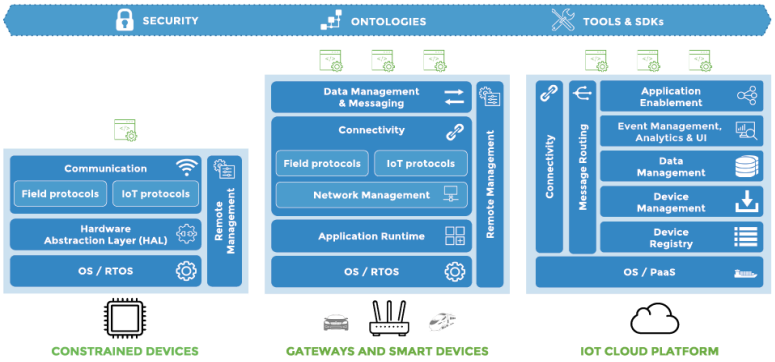
1. Seguridad: la seguridad debe implementarse desde los dispositivos hasta la nube. Funciones como autenticación, cifrado y autorización deben formar parte de cada pila.

2. Ontologías: el formato y la descripción de los datos del dispositivo es una característica importante para permitir el análisis de datos y la interoperabilidad de datos. La capacidad de definir ontologías y metadatos en dominios heterogéneos es un área clave para IoT.

3. Herramientas de desarrollo y SDK: los desarrolladores de IoT requerirán herramientas de desarrollo que admitan las diferentes plataformas de hardware y software involucradas.

Figura 10

Arquitectura de una solución IoT estándar



Nota. Fuente. Eclipse Foundation (2019)

Características clave de los stack de IoT

1. Acoplado libremente: se han definido tres pilas de IoT, pero es importante que cada pila se pueda usar independientemente de las otras pilas. Debería ser posible utilizar una plataforma en la nube de IoT de un proveedor con una puerta de enlace de IoT de otro proveedor y un tercer proveedor para la pila de dispositivos.

2. Modular: cada pila debe permitir que las características se obtengan de diferentes proveedores.

3. Independiente de la plataforma: cada pila debe ser independiente del hardware del host y de la infraestructura de la nube. Por ejemplo, la pila de dispositivos debe estar disponible en múltiples MCU y la plataforma en la nube de IoT debe ejecutarse en diferentes Cloud PaaS.

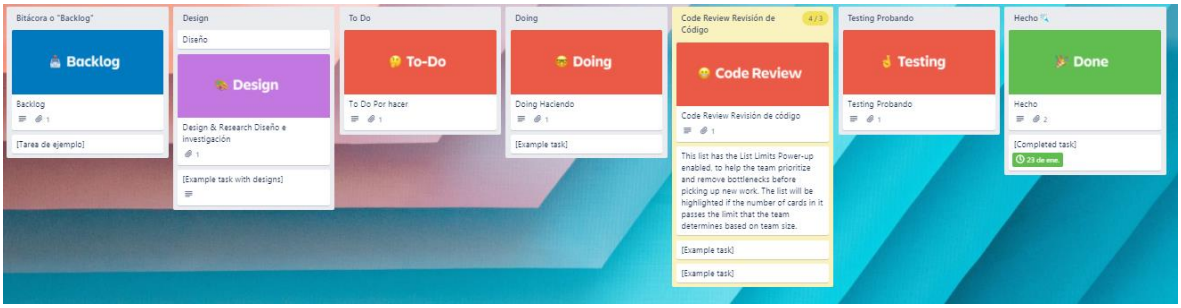
4. Basado en estándares abiertos: la comunicación entre las pilas debe basarse en estándares abiertos para garantizar la interoperabilidad.

5. API definidas: cada pila debe tener API definidas que permitan una fácil integración con las aplicaciones existentes y la integración con otras soluciones de IoT.

En cuanto a metodologías, Kanban es una con la que se gestiona y planifica el trabajo de una manera muy sencilla y gráfica. Aportando practicidad y eficiencia en este proceso mientras se visualizan las tareas que se deben llevar a cabo y el proceso por el que estas están pasando como se visualiza en la figura 11.

Figura 11

Ejemplo de distribución de tareas y actividades en Kanban



Nota. Fuente. Creación propia (2020)

Algunos de los beneficios de esta metodología son la constante mejora en el flujo de las actividades y la calidad de estas, ya que limita y evita la acumulación de trabajo. Si es cierto que no es la metodología más empleada para el desarrollo de proyectos, es la ideal para este en particular, dada la situación actual y el tamaño del proyecto. Teniendo en cuenta factores como:

- Cadencia de flujo continuo.
- Metodología de entrega continua.
- No hay funciones establecidas, por lo que es ideal para grupos pequeños.
- Filosofía de cambio versátil, se puede producir en cualquier momento.

En cuanto a los dispositivos electrónicos a utilizar en el proyecto se tienen los siguientes:

Arduino. Se puede definir como plataforma electrónica opensource cuya principal característica es el uso fácil en cuanto al software y hardware. Adicionalmente pueden leer entradas análogas y digitales para su procesamiento y posterior interacción con otros dispositivos

tales como motores LED’s, tableros, etc. Su funcionamiento, en líneas generales, inicia indicando a la placa qué hacer a través de una serie de instrucciones al microcontrolador. Para ello, la plataforma un el lenguaje de programación Arduino (basado en Wiring) y cuenta con un IDE propia basado en Processing sin embargo se pueden usar otros IDE como Visual Studio Code que, a través de plugins, proveen la funcionalidad de desarrollo para Arduino.

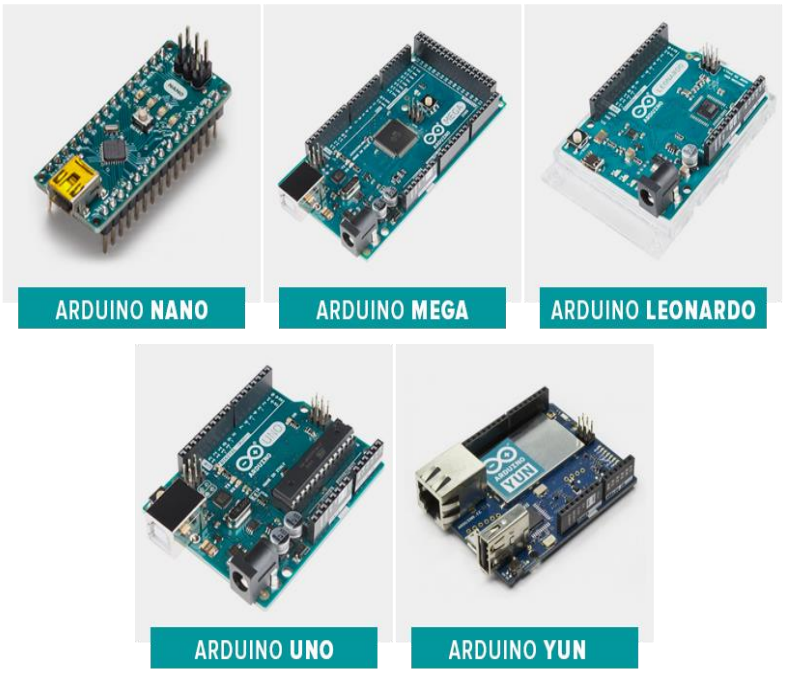
Al ser opensource, Arduino ha sido utilizado en miles de proyectos de diferentes calibres, desde de uso cotidiano hasta complejos instrumentos industriales y científicos. Las contribuciones a la plataforma provienen de una comunidad mundial de diferentes perfiles como estudiantes, aficionados, desarrolladores de software, etc. quienes han generado una increíble cantidad de conocimiento el cual puede ser accedido de forma libre por principiantes y expertos en sus proyectos.

Arduino fue producto de una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación de software en el Ivrea Interaction Design Institute. Resultó sen tan versatilidad que, al llegar a una comunidad mayor, la placa Arduino inicio un proceso de adaptación a los nuevos desafíos, lo cual es un gran diferencial en su oferta desde placas de 8 bits hasta productos relacionados con de IoT, wearables, impresión 3D y entornos integrados.

La plataforma electrónica Arduino, al ser de código abierto, permite a los usuarios construirlas y adaptarlas a sus necesidades sin que ello implique inconvenientes de derechos de autor. En cuanto al software, gracias a una comunidad entusiasta que va en aumento, se han creado gran variedad de librerías y características disponibles de forma libre acordes con la evolución de los proyectos en los cuales se ha utilizado Arduino. En la figura 12 se pueden apreciar las diferentes versiones

Figura 12

Versiones de Arduino. Arduino



Nota. Fuente: Arduino.cl (2018)

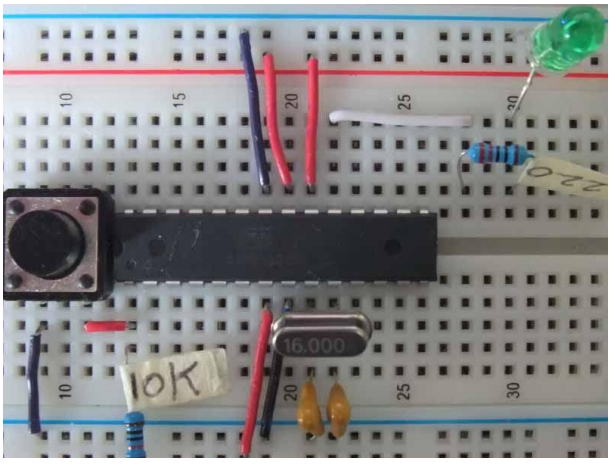
Si bien existen otras propuestas similares como BASIC Stamp de Parallax, Netmedia' s BX-24, Phidgets, MIT ' Handyboard s, Arduino ofrece algunas ventajas para profesores, estudiantes y aficionados interesados sobre otros sistemas:

- Bajo costo: La versión de Arduino para su ensamblaje a mano y sus módulos preensamblados cuestan menos de US$50 lo cual representa un costo bajo en comparación con otras plataformas electrónicas licenciadas.
- Multiplataforma - El software Arduino se puede ejecutar en Windows, Macintosh OS X, y Linux.
- IDE simple: es fácil es fácil de usar para principiantes, pero lo suficientemente flexible para que los usuarios avanzados también lo aprovechen e incluso creen productos para su comercialización.

- Código Opensource: el software Arduino al ser de código abierto está disponible para su extensión por programadores experimentados. El lenguaje se puede ampliar a través de librerías C ++, y las personas que quieren entender los detalles técnicos pueden dar el salto de Arduino a lenguaje AVR-C en la cual se basa agregando código en dicho lenguaje en sus programas Arduino.
- Hardware de código abierto y extensible: Con esquemas publicados bajo licencia Creative Commons, los diseñadores de circuitos experimentados pueden desarrollar versiones propias del módulo, ampliar funcionalidades e incluso mejorar las ya existentes. Usuarios con poca experiencia están en la capacidad de desarrollar una versión de la placa en una protoboard como se muestra en la figura 13a fin de comprender su funciona reducir costos en componentes electrónicos.

Figura 13

Esquema Arduino implementado en una protoboard



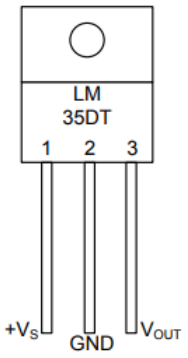
Nota. Fuente: geekfactory.com (2008)

LM35: es un sensor digital, cuyo esquema se visualiza en la figura 14, que nos permitirá monitorear la temperatura del cultivo durante su crecimiento, este sensor nos provee un análisis

de temperatura lineal que va desde los -55° y los 150° Celsius. A diferencia de los termistores convencionales que nos proporciona una variación en la resistividad eléctrica conforme varía la temperatura, el LM35 proporciona una variación de voltaje, lo cual nos facilita su interpretación a la hora de digitalizar estas señales.

Figura 14

Disposición de integrado LM35



Nota. Fuente: Texas Instruments LM35 datasheet (2017)

En la figura 15 se relacionan algunos datos proporcionados por su fabricante “Texas Instruments”, características que nos permiten analizar variables como su exactitud, umbral de análisis y rangos de no linealidad.

Tabla 1

Características integrado LM35

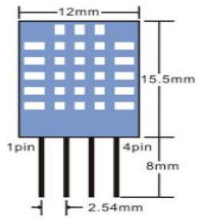
PARAMETER	***TEST CONDITIONS***	***LM35A***			***LM35CA***			***UNITS (MAX.)***
		TYP	***TESTED LIMIT***	***DESIGN LIMIT***	***TYP***	***TESTED LIMIT***	***DESIGN LIMIT***	
Accuracy	$T_A = 25ºC$	*±0.2*	*±0.5*		*±0.2*	*±0.5*		*°C*
	$T_A = -10ºC$	*±0.3*			*±0.3*		*±1*	
	$T_A = T_{MAX}$	*±0.4*	*±1*		*±0.4*	*±1*		
	$T_A = T_{MIN}$	*±0.4*	*±1*		*±0.4*		*±1.5*	
Nonlinearity	$T_{MIN} \leq T_A \leq T_{MAX}$	*±0.18*		*±0.35*	*±0.15*		*±0.3*	*°C*
Sensor gain (average slope)	$T_{MIN} \leq T_A \leq T_{MAX}$	*+10*	*+9.9* *+10.1*		*+10*		*+9.9* *+10.1*	*mV/°C*
Load regulation	$T_A = 25ºC$	*±0.4*	*±1*		*±0.4*	*±1*		*mV/mA*
	$T_{MIN} \leq T_A \leq T_{MAX}$	*±0.5*		*±3*	*±0.5*		*±3*	
Line regulation	$T_A = 25ºC$	*±0.01*	*±0.05*		*±0.01*	*±0.05*		*mV/V*
	$4V \leq V_S \leq 30\,V$	*±0.02*		*±0.1*	*±0.02*		*±0.1*	

Nota. Fuente: Texas Instruments LM35 datasheet (2017)

El DHT11es un sensor cuya disposición se visualiza en la figura 16 que nos permitirá monitorear la temperatura ambiente y la humedad relativa en conjunto, uno de los beneficios de este integrado, es su estabilidad a largo plazo, además de que nos proporcionará una alta confiabilidad en cuanto a sus datos, adquiridos por medio de señales digitales fácilmente analizables por cualquier tipo de microprocesador. A diferencia del LM35, este nos permitirá detectar, además de la humedad relativa, la temperatura ambiente con mayor precisión, ya que el LM35 es un dispositivo recomendado para suelo o superficies.

Figura 15

Dispositivo integrado DHT11



Nota. Fuente: Texas instruments DHT11 Datasheet (2010)

En cuanto a sus especificaciones técnicas, y basado en los datos proporcionados por su distribuidor “Mouser Electronics”, su exactitud maneja rangos de ±4% en cuanto a la medición de humedad relativa y ±1°C en la medición de temperatura, y una resolución de 1% y 0.1°C respectivamente. La figura 17 se pueden apreciar las demás características

Figura 16

Características DHT11

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S

Nota. Fuente: Texas instruments DHT11 Datasheet (2010)

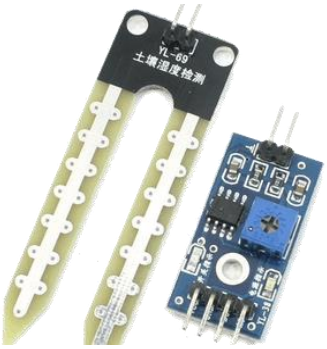
Para censar la humedad del suelo, tenemos en consideración dos posibles soluciones. La primera es un sensor convencional para este tipo de mediciones, y la segunda es una disposición diseñada por nosotros que cubra todo el masetero internamente. A continuación, se exponen las características, ventajas y desventajas de cada uno.

Los sensores YL 38 - YL 69 permiten detectar y monitorear la humedad del suelo. Al igual que los sensores digitales que usaremos para los demás parámetros, estos proporcionan una salida de voltaje que varía de acuerdo con el parámetro analizado, en este caso, la humedad.

- Ventaja: Una de las ventajas que nos ofrece este sensor es la facilidad de instalación y mantenimiento, ya que al ser un integrado, basta con reemplazarlo en caso de daño y no se tendrá que modificar de ninguna manera el código del sistema microprocesador.
- Desventaja: A pesar de que el sensor maneja una exactitud y resolución válida para llevar a cabo esta medición en nuestro dispositivo, su disposición en el macetero dependerá de lo que mida, ya que, como sabemos, la humedad no siempre se distribuye uniformemente por todo el terreno, por lo que este sensor puede detectar valores sólo en su área de medición y no en toda la disposición del macetero.

Figura 17

Sensores YL 38 - YL 69

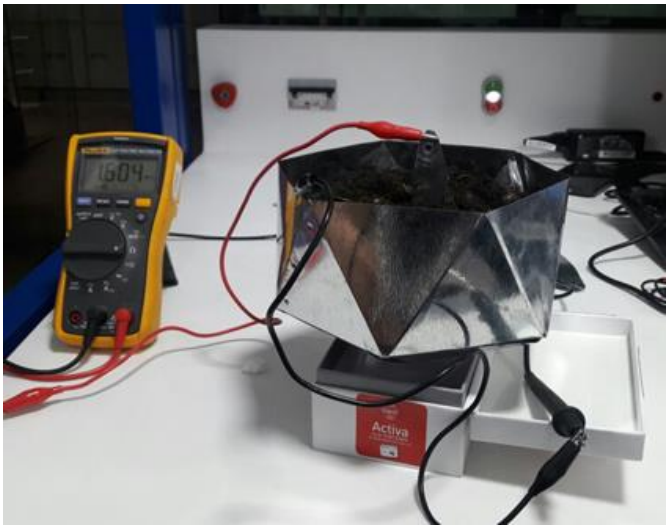


Nota. Fuente: Electrónicos Caldas (2020)

Otro sensor de humedad de carácter experimental se despliega en macetero. Se propone entonces una solución para solucionar el problema de la exactitud en la medición de la humedad de todo el macetero, el cual es un recipiente metálico cuyo valor de capacitancia variará conforme varíe su contenido.

Figura 18

Prototipo en puesta a prueba



Nota. Fuente: Elaboración propia (2020)

Podemos observar que, a partir de un recipiente metálico y una pieza de acero, que actúan como electrodos, se analizan diferentes valores capacitivos que varían conforme se reemplace su componente interior, teniendo en cuenta sus valores de resistencias dieléctricas a una temperatura ambiente.

- Ventajas: Solucionamos el inconveniente de no tener un valor de humedad general en el macetero, siendo que los valores que detecte serán dependiendo del contenido en general. Además, que es un sensor que puede soportar cadencias elevadas de funcionamiento continuo, característica ideal para un dispositivo que será usado por usuarios que desconocen su funcionamiento.

- Desventajas: Siendo un sensor experimental que funciona a partir de valores capacitivos, requiere de un circuito equivalente, el cual interprete estos valores en función de voltaje y los linealice en rangos de medición coherentes. Además, su sensibilidad es elevada, lo que puede hacer que arroje valores incoherentes si detecta materiales distintos al metal, como polvo, o materiales granulados.

En cuanto al sistema de control, un STM32F407VG, fabricado por STMelectronics, cuenta con un acelerador adaptativo en tiempo real, una frecuencia de trabajo de hasta 168MHz y la capacidad de realizar procesos embebidos según se requiera. Algunas de las características con las que cuenta este núcleo son:

- Capacidad para realizar operación de bajo poder
- Gestión de reloj, reinicio y suministro.
- Interfaz paralela LCD.
- Conectividad USB 2.0 full-speed
- Variedad de puertos para adquisición de señales análogas y digitales.

Figura 18. Microprocesador STM32F407VG.

Figura 19

Microprocesador STM32F407VG.

Nota. Fuente: STMicroelectornics (2020)

Su método de programación es a partir de un sistema operativo en tiempo real (RTOS), lo cual proporciona funciones de multitarea, teniendo en cuenta el tiempo de cada una a realizarse. Es un sistema de control bastante completo.

Como todo sistema de naturaleza informática, más aún si va a estar expuesto en internet, el presente proyecto plantea una serie de retos relacionados con amenazas y vulnerabilidades que deben ser analizadas.

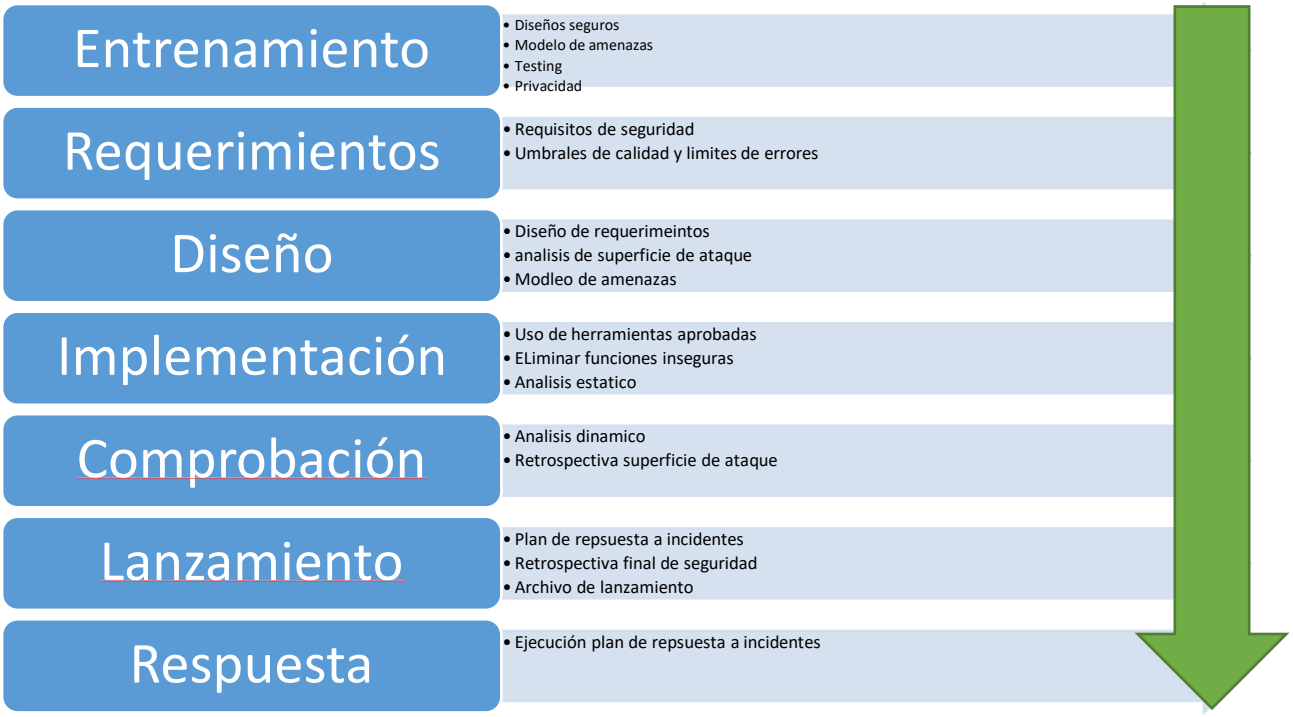
Al respecto, Microsoft ha desarrollado un proceso un proceso de garantía de seguridad denominado SDL, por sus siglas en ingles Security Development Lifecycle) que se centra en el desarrollo de software cuyo como objetivo es reducir el número y la gravedad de las vulnerabilidades en el software donde se introduce seguridad y privacidad en todas las fases del proceso de desarrollo.

Microsoft SDL se basa en tres conceptos principales: educación, mejora continua de procesos y rendición de cuentas. Para los roles técnicos de un grupo de desarrollo de software es fundamental la educación y capacitación continúa realizando una inversión adecuada en transferencia de conocimiento que permite a reaccionar a los equipos y organizaciones de forma adecuada a cambios en la tecnología y/o escenarios de amenaza. Como el riesgo de seguridad es dinámico, SDL busca comprender la causa y el efecto de las vulnerabilidades de forma constante por lo cual requiere una evaluación periódica de los procesos del ciclo de vida de desarrollo seguro y la integración de cambios dados a nuevos avances tecnológicos amenazas. Los datos se recopilan para evaluar la eficacia de la capacitación, las métricas en curso se utilizan para confirmar el cumplimiento del proceso y las métricas posteriores al lanzamiento ayudan a guiar los cambios futuros. Por último, el SDL requiere el archivo de todos los datos necesarios para dar servicio a una aplicación en crisis. Cuando se combina con planes detallados de respuesta y

comunicación de seguridad, una organización puede proporcionar orientación concisa y convincente a todas las partes afectadas.

Figura 20

Modelo SDL



Nota. Fuente: Elaboración propia (2021)

Metodología

La metodología de trabajo que se abordó adoptó para el desarrollo de este proyecto es Kanban dada la naturaleza del proyecto donde se tienen dos aspectos bien diferenciados, el software y el hardware donde cada uno de los integrantes del grupo de trabajo tiene habilidades especificas en uno de esos aspectos lo cual conlleva a una diferenciación de tareas que pueden ser gestionadas de forma más ágil y precisa con esta metodología.

Adicionalmente, dadas las actuales restricciones de movilidad del año 2020 a cuenta de la pandemia, consideramos que la metodología Kanban a través de las diferentes herramientas existentes en el mercado, facilita la organización de trabajo, para la aplicación de esta se determinan los siguientes acuerdos iniciales:

Estados: Para el manejo del flujo de proceso se establecen para el presente proyecto los siguientes:

- Backlog: Pila de tareas que se deben realizar durante todo el proyecto
- To Do: Las tareas próximas a realizar, a diferencia del backlog este estado de cumplir los acuerdos de máximos determinados por el equipo de los cuales se mencionan más adelante
- Doing: Las tareas que están en proceso
- Review: revisión de la tarea
- Done: Tarea terminada, esta se define desde los criterios de aceptación

Priorización, se manejarán 4 prioridades inicialmente:

- Estándar: Tareas que no tienen alto impacto en el proyecto y que no son prerrequisitos de otras tareas

- Importante: Tareas prerrequisitos, de no realizarse impactan el desarrollo del proyecto
- Urgente: Tareas que pueden llegar a detener el proyecto de no realizarse o tienen mucha importancia para el desarrollo de otras tareas
- Trabajo no planificado: Tareas que surgen durante el desarrollo del proyecto

Adicionalmente se determina los límites del trabajo en proceso o WIP (por sus siglas en inglés work in progress) donde se definen la cantidad de tareas máxima que puede tener un estado a fin detectar problemas de desarrollo para su priorización, para este proyecto se definen:

- Máximo 6 actividades total
- Máximo 2 actividades urgentes
- Máximo 3 actividades importantes

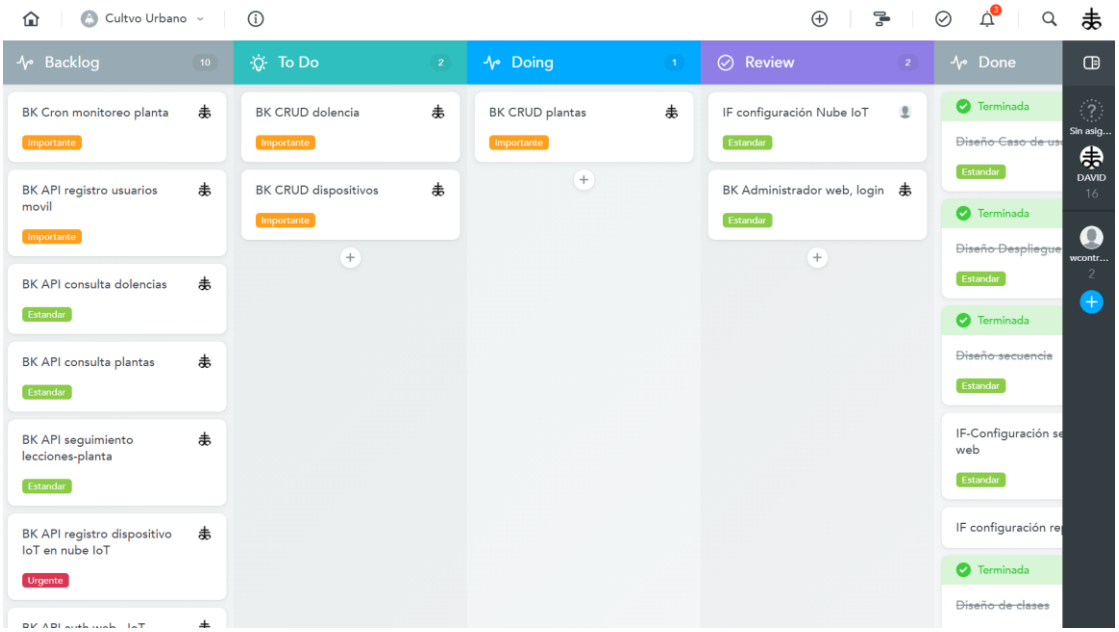
También se define una nomenclatura pues el proyecto plantea 4 componentes principales interrelacionados: dispositivos sensores, gateway, nube y móvil por lo cual se determinan una tipificación de tareas identificada con la siguiente nomenclatura para identificar el campo de acción de la tarea dentro de la solución general

- HW: tareas relacionadas con el hardware de captura de datos o sensores
- BK: tareas relacionadas con el backend de la solución
- App: tareas relacionadas con el desarrollo móvil
- GW: tareas desarrolladas con el gateway de la solución

En la figura 21 se pueden visualizar la aplicación de la nomenclatura mencionada anteriormente.

Figura 21

Tablero Kanban del proyecto



Nota. Fuente: Elaboración propia (2021).

Kanban permitió en durante el desarrollo del proyecto, llevar un control y consulta del estado del proyecto y conocer compartir conocimiento, a partir de las tareas enfocadas den los componentes de hardware y software para los integrantes del grupo. Adicionalmente, debido a la definición del límite de tareas se pudieron detectar inconvenientes en el desarrollo del proyecto al determinar las tareas que supusieron mayor esfuerzo para su realización.

Por otra parte, permitió una distribución de tareas uniforme a partir de los conocimientos de los integrantes del grupo de trabajo agilizando el desarrollo del proyecto, paralelamente también se definieron tares de diseño y arquitectura que se desarrollaron a través de herramientas colaborativa como lucid chart y g-suite.

Esta administración de tareas realizado a través de la metodología Kanban fueron gestionadas de acuerdo con el proceso de software que se describe a continuación.

Proceso de software

A fin de crear la solución acorde a los objetivos de este proyecto, se aplicó los conceptos y herramientas que ofrecen las diferentes etapas del ciclo de vida de desarrollo de software ejecutada bajo la metodología Kanban.

7.1. Requerimientos funcionales

Estos requerimientos son los asociados a las funciones que el sistema debe poseer para satisfacer las necesidades del usuario o como lo define Sommerville (2005) como "Son declaraciones de los servicios que debe proporcionar el sistema, de la manera que este debe reaccionar a entradas particulares y de cómo se de comportar en situaciones particulares. En algunos casos, los requerimientos no funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer"

Los requerimientos funcionales expresados en escenarios operacionales brindan información contextual de la funcionalidad para determinar viabilidad y/o prioridad. A continuación, se relacionan los requerimientos funcionales relevantes en las tablas 1 a 9.

Tabla 2

Escenario operacional 1, consultar planta.

Escenario Operacional:	Consultar planta.
Stakeholder Asociado	Usuario
Descripción general de la funcionalidad	El usuario se autentica y puede consultar desde su dispositivo móvil, a partir de una lista que entrega el sistema, la patología sobre la cual quiere consultar la(s) planta(s) que sirven de tratamiento para la misma, posteriormente escoge la planta de su interés y visualiza las instrucciones de cultivo.
Describa lo que el Stakeholder hace ahora o le gustaría poder hacer	Actualmente, a través de páginas web u otras aplicaciones consulta la planta que necesita para tratar la patología y luego, en otras fuentes o aplicaciones consulta cómo cultivar dicha planta. LE gustaría realizar tales consultas desde una única fuente.
Describa cualquier entrada provista o disponible al momento del inicio	Se debe tener parametrizado en el sistema el listado de patologías tratables con plantas medicinales y con las plantas asociadas a cada patología y las instrucciones de cultivo.

Describa el contexto de la operación	La operación se realiza a petición del usuario.
Describa cómo el sistema debe responder	El sistema a partir de listas desplegables debe mostrar de forma clara las plantas que puede usar para tratar la patología escogida inicialmente y, a través de un “ver más...” las instrucciones para el cultivo.
Describa las salidas que el sistema produce como resultado de la acción	Ninguna.
Describa quién o qué usa la salida y para qué es utilizada	No aplica.

Nota. Fuente: Elaboración propia (2021)

Tabla 3

Escenario operacional 2, Registrar dispositivo.

Título del Escenario Operacional:	Registra dispositivo
Stakeholder Asociado	Usuario
Descripción general de la funcionalidad	El usuario se autentica y puede registrar en el sistema su Asistente de cultivo IoT.
Describa lo que el Stakeholder hace ahora o le gustaría poder hacer	Actualmente no cuenta con un dispositivo sensor que mida las condiciones de la planta en tiempo real. Le gustaría consultarlo con un clic desde su dispositivo móvil.
Describa cualquier entrada provista o disponible al momento del inicio	Número de serie del dispositivo provisto por el usuario, dicho serial debe estar en el sistema para su validación de registros.
Describa el contexto de la operación	La operación se realiza a petición del usuario.
Describa cómo el sistema debe responder	Con el serial valida la existencia del dispositivo, asocia el dispositivo al perfil de usuario y lo activa en la nube IoT AWS.
Describa las salidas que el sistema produce como resultado de la acción	El sistema debe confirmar al usuario a través de email el registro del dispositivo, adicionalmente enviar un mensaje de confirmación al sensor.
Describa quién o qué usa la salida y para qué es utilizado	El usuario, una vez confirmado el registro del Asistente de cultivo IoT, debe asociar la planta de su interés a su biblioteca para posterior consulta.

Nota. Fuente: Elaboración propia (2021)

Tabla 4
Escenario operacional 3, agregar planta a la biblioteca.

Título del Escenario Operacional:	Agregar planta a la biblioteca
Stakeholder Asociado	Usuario
Descripción general de la funcionalidad	El usuario puede adicionar plantas a su biblioteca para posterior consulta del estado de esta.
Describa lo que el Stakeholder hace ahora o le gustaría poder hacer	Le gustaría tener un listado de sus plantas de interés sobre las cuales va a utilizar el Asistente de cultivo IoT.
Describa cualquier entrada provista o disponible al momento del inicio	Tener mínimo un Asistente de Cultivo IoT registrado en el sistema.
Describa el contexto de la operación	La operación se realiza a petición del usuario.
Describa cómo el sistema debe responder	Consulta de prueba para validar si el Asistente de Cultivo IoT está en funcionamiento, si lo está, agrega al perfil del usuario la planta seleccionada.
Describa las salidas que el sistema produce como resultado de la acción	Si el Asistente de Cultivo IoT no responde se notifica al usuario a través de la aplicación móvil.
Describa quién o qué usa la salida y para qué es utilizado	EL usuario valida si el Asistente de IoT está encendido y conectado a la red.

Nota. Fuente: Elaboración propia (2021)

Tabla 5
Escenario operacional 4, consultar Estado de la planta

Título del Escenario Operacional:	Consultar Estado de la planta.
Stakeholder Asociado	Usuario
Descripción general de la funcionalidad	El usuario se autentica y puede consultar el estado de la planta.
Describa lo que el Stakeholder hace ahora o le gustaría poder hacer	Actualmente no cuenta con una aplicación móvil que permita consultar el estado de la planta. Le gustaría consultarlo seleccionando la planta de su interés registrada como “en cultivo” en el sistema.
Describa cualquier entrada provista o disponible al momento del inicio	Registros de datos sobre temperatura, humedad y pH de la planta generados por el asistente de cultivo IoT en las últimas 24 horas en la nube AWS IoT.

Describa el contexto de la operación	La operación se realiza a petición del usuario.
Describa cómo el sistema debe responder	El sistema valida si existen datos registrados por el Asistente de Cultivo IoT sobre la planta seleccionada por el usuario y obtiene el último valor registrado el cual se envía a la aplicación móvil para su visualización.
Describa las salidas que el sistema produce como resultado de la acción	No hay salidas.
Describa quién o qué usa la salida y para qué es utilizada	No aplica.

Nota. Fuente: Elaboración propia (2021)

Tabla 6

Escenario operacional 5, alerta estado planta

Título del Escenario Operacional:	Alerta estado planta.
Stakeholder Asociado	Usuario
Descripción general de la funcionalidad	Monitorear cada hora los valores de pH, humedad y/o temperatura de las plantas registradas.
Describa lo que el Stakeholder hace ahora o le gustaría poder hacer	En caso de que los valores de pH, humedad y /o temperatura estén por fuera de los límites establecidos en el sistema al usuario le gustaría ser notificado en su dispositivo móvil.
Describa cualquier entrada provista o disponible al momento del inicio	La planta debe tener definido en el sistema sus valores máximos y mínimos de pH, temperatura y humedad.
Describa el contexto de la operación	Es una tarea programa cada hora donde se obtienen y validan los datos de la planta.
Describa cómo el sistema debe responder	Una vez detectado un valor fuera de límite establecido debe enviar una notificación push al usuario indicando la novedad.
Describa las salidas que el sistema produce como resultado de la acción	No hay salidas.

Describa quién o qué usa la salida y para qué es utilizada	No aplica.

Nota. Fuente: Elaboración propia (2021)

Tabla 7

Escenario operacional 6, alerta estado sensor.

Título del Escenario Operacional:	Alerta estado sensor.
Stakeholder Asociado	Usuario
Descripción general de la funcionalidad	Monitorear cada hora el estado del Asistente de IoT, se valida si es detectable y midiendo datos de la planta.
Describa lo que el Stakeholder hace ahora o le gustaría poder hacer	En caso de que el sensor no funcione, el usuario le gustaría ser notificado en su dispositivo móvil sobre esta novedad.
Describa cualquier entrada provista o disponible al momento del inicio	Dispositivo IoT registrado en el sistema.
Describa el contexto de la operación	Es una tarea programa cada hora donde se obtiene el estado del sensor.
Describa cómo el sistema debe responder	Una vez no se detecte un Asistente de IoT al consultar datos de una planta, o la hora del último dato registrado de una planta sea superior a 1 a partir del momento de la consulta debe enviar una notificación push al usuario indicando la novedad.
Describa las salidas que el sistema produce como resultado de la acción	No hay salidas.
Describa quién o qué usa la salida y para qué es utilizada	No aplica.

Nota. Fuente: Elaboración propia (2021)

Tabla 8

Escenario operacional 7, CRUD patología

Título del Escenario Operacional:	CRUD sobre patología
Stakeholder Asociado	Administrador
Descripción general de la funcionalidad	Operaciones CRUD en las patologías del sistema.
Describa lo que el Stakeholder hace ahora o le gustaría poder hacer	Al administrador le gustaría tener una interfaz muy intuitiva para registrar, consultar, actualizar o desactivar las patologías que los usuarios del sistema podrán consultar.
Describa cualquier entrada provista o disponible al momento del inicio	Usuarios administradores registrados en el sistema.
Describa el contexto de la operación	La operación se realiza a petición del administrador.
Describa cómo el sistema debe responder	Actualizar los registros de patologías en el sistema informando al administrador tal actualización.
Describa las salidas que el sistema produce como resultado de la acción	No hay salidas.
Describa quién o qué usa la salida y para qué es utilizada	No aplica.

Nota. Fuente: Elaboración propia (2021)

Tabla 9

Escenario operacional 8, CRUD plantas

Título del Escenario Operacional:	CRUD sobre plantas
Stakeholder Asociado	Administrador

Descripción general de la funcionalidad	Operaciones CRUD en las plantas del sistema
Describa lo que el Stakeholder hace ahora o le gustaría poder hacer	Al administrador le gustaría tener una interfaz muy intuitiva para registrar, consultar, actualizar o desactivar las plantas que los usuarios del sistema podrán consultar
Describa cualquier entrada provista o disponible al momento del inicio	Usuarios administradores registrados en el sistema, patologías registradas en el sistema
Describa el contexto de la operación	La operación se realiza a petición del administrador
Describa cómo el sistema debe responder	Actualizar los registros de las plantas en el sistema asociados a una patología determinada informando al administrador tal actualización
Describa las salidas que el sistema produce como resultado de la acción	Lista de plantas actualizada en las aplicaciones móviles del sistema y notificación push sobre la actualización de esta información a los usuarios finales
Describa quién o qué usa la salida y para qué es utilizada	Los usuarios finales podrán seleccionar una patología y consultar y/o agregar a sus bibliotecas la planta de interés para su monitoreo

Nota. Fuente: Elaboración propia (2021)

Tabla 10

Escenario operacional 9, registro de Asistentes de Cultivo IoT

Título del Escenario Operacional:	Registro de Asistentes de Cultivo IoT
Stakeholder Asociado	Administrador
Descripción general de la funcionalidad	Registrar en el sistema los seriales de los Asistentes de Cultivo IoT a comercializar
Describa lo que el Stakeholder hace ahora o le gustaría poder hacer	Al administrador le gustaría tener una interfaz muy intuitiva para registrar, consultar, actualizar o desactivar Asistentes de Cultivo IoT
Describa cualquier entrada provista o disponible al momento del inicio	Usuarios administradores registrados en el sistema.

Describa el contexto de la operación	La operación se realiza a petición del administrador
Describa cómo el sistema debe responder	Actualizar los registros de Asistente de Cultivo IoT en el sistema
Describa las salidas que el sistema produce como resultado de la acción	Lista de Asistente de Cultivo IoT actualizada
Describa quién o qué usa la salida y para qué es utilizada	Los usuarios finales podrán registrar sus Asistentes de Cultivo adquiridos para el uso de la plataforma a partir de un número de serie.

Nota. Fuente: Elaboración propia (2021)

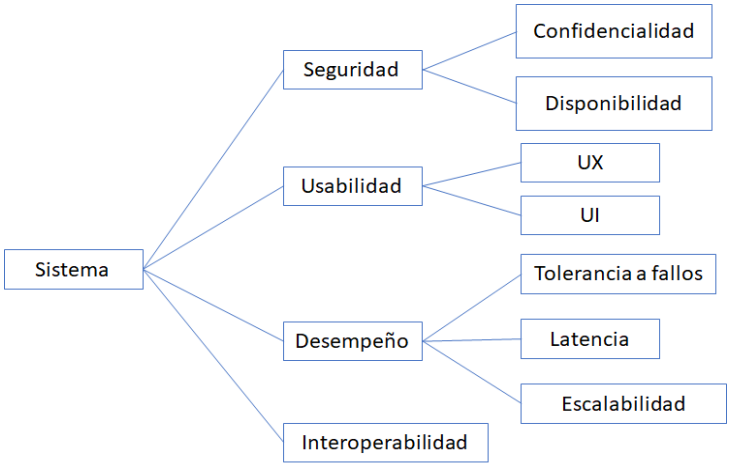
7.2. Requerimientos no funcionales

Presentan restricciones de los servicios o funciones del sistema como tiempo, proceso de desarrollo y estándares, por lo general se aplican a todo el sistema y, a diferencia de las funcionales, apenas se aplican sobre servicios individuales o funcionalidades del sistema.

Para lograr una mayor comprensión, se relaciona el árbol de utilidad a fin de visualizar el relacionamiento de los atributos de calidad del producto.

Figura 22

Árbol de utilidad



Nota. Fuente: Elaboración propia (2021)

A continuación, en las tablas 10 a 16, se detalla los requerimientos no funcionales enmarcados en los atributos de calidad definidos para el proyecto, los cuales se consideran en el diseño de la solución para el aprendizaje, implementación y seguimiento de cultivos urbanos medicinales relacionando su prioridad a partir de los intereses de Stakeholders, restricciones de desarrollo y la naturaleza del proyecto

Tabla 11

Requerimiento no funciona 1, confidencialidad.

Requerimiento no funcional	Seguridad	Confidencialidad	Prioridad
Autorización	En el componente móvil solo usuarios registrados pueden acceder al sistema		Alta
Autorización	En el componente Web, solo los usuarios administradores deben acceder al sistema		Alta
Autorización	En el componente de IoT, solo dispositivos registrados previamente pueden acceder al servicio AWS IoT		Alta

Nota. Fuente: Elaboración propia (2021)

Tabla 12

Requerimiento no funcional 2, disponibilidad

Requerimiento no funcional	Seguridad	Disponibilidad	Prioridad
	El componente web debe estar disponible el 99% de tiempo al año		Alta
	En el componente de IoT, el sistema debe estar disponible el 80% de tiempo al año en lapsos no mayores de una hora fuera de línea		Media

Nota. Fuente: Elaboración propia (2021)

Tabla 13

Requerimiento no funcional 3, usabilidad UI

Requerimiento no funcional	Usabilidad	UI	Prioridad
Línea gráfica	Todas las interfaces de usuario del sistema (móvil, administración y dispositivo IoT) deben atender una misma línea gráfica que mantenga un diseño limpio de interfaces.		Baja
Notificaciones	Los mensajes en las notificaciones push, en el componente móvil, deben ser concisos para su visualización completa en la notificación del dispositivo móvil		Baja

Nota. Fuente: Elaboración propia (2021)

Tabla 14

Requerimiento no funciona 4, usabilidad UX.

Requerimiento no funcional	Usabilidad	UX	Prioridad
Interacciones	En el componente móvil, se debe mantener la simplicidad en la(s) petición(es) a través de un mínimo de interacciones manteniendo solo una vista por cada funcionalidad		Alta
Interacciones	En el componente IoT, la interacción con el usuario final debe ser solo de fines informativos salvo la configuración de WI FI		Media
Interacciones Flujo de datos	En el componente web, las funciones administrativas deben mantener e indicar el flujo de tareas para determinada acción en la misma vista sin actualizar el navegador		Baja

Nota. Fuente: Elaboración propia (2021)

Tabla 15

Requerimiento no funcional 6, latencia.

Requerimiento no funcional	Desempeño	Latencia	Prioridad
	El sistema, en su componente móvil, debe realizar la consulta de datos y obtener su respuesta en 2 segundos desde el momento de activar el botón de consulta. El registro de fallas o alertas al usuario relacionados con el Asistente de Cultivo IoT debe tardar máximo 2 segundos desde el momento en que se detecta la novedad. El registro de fallas relacionados con la consulta a la nube IoT deben ser notificadas al administrador del sistema y a los usuarios en, al menos, 2 segundo.		Alta

Nota. Fuente: Elaboración propia (2021)

Tabla 16

Requerimiento no funcional 7, tolerancia a fallos.

Requerimiento no funcional	Desempeño	Tolerancia a fallos	Prioridad
	Detectar cada 30 minutos el estado de los Asistente de cultivo IoT, en caso de desconexión notificar al usuario para que verifique Consultar la conexión de forma constante de la plataforma a la nube de IoT, en caso de desconexión notificar a los usuarios de forma inmediata		Alta

Nota. Fuente: Elaboración propia (2021)

Tabla 17

Requerimiento no funcional 6, interoperabilidad.

Requerimiento no funcional	Interoperabilidad		Prioridad
	En el componente móvil, la aplicación funcionará sobre sistemas operativos iOS 8 y Android 7 (Nougat) en adelante. El componente web, debe consultarse desde navegadores web de escritorio y móviles Chrome 88.0.4324.150 y Mozilla Firefox 52.6 en adelante.		Alta

Nota. Fuente: Elaboración propia (2021)

Tabla 18 Requerimiento no funcional 7, mantenibilidad

Requerimiento no funcional	Mantenibilidad		Prioridad
	El sistema de información debe permitir agregar nuevas funcionalidades por otros desarrolladores de forma ágil		Media

Nota. Fuente: Elaboración propia (2021)

Si bien los requerimientos no funcionales mencionados anteriormente plantean como mínimo 8 escenarios de calidad a continuación, para este proyecto se evaluarán 3 relacionados en las tablas en las tablas 19 a 21 los cuales son los más relevantes para un prototipo a fin de caracterizar y capturar aspectos de atributos de calidad (QAs) de una forma concreta, que puede ser evaluada y utilizada en las actividades de diseño, y luego durante el desarrollo.

Tabla 19

Escenario de calidad 1, latencia.

Requerimiento no funcional	Latencia	Prioridad	Alta
Justificación	El grupo de desarrollo establece la importancia de ofrecer una respuesta rápida al usuario para posicionar una imagen de agilidad y eficiencia en el producto para un posible emprendimiento derivado del proyecto.		

Fuente	Usuario Final.
Estímulo	El usuario consulta el estado de la planta.
Artefacto	Aplicación móvil, Componente Web, Nube IoT, dispositivo IoT.
Ambiente	Operación normal del componente web y los servicios de Thinker.io
Respuesta	El usuario ve la información solicitada en entre 2 y 4 segundos.
Medida de la respuesta	El 90 % de las peticiones deben realizarse en un lapso no mayor a los 3 segundos.

Nota. Fuente: Elaboración propia (2021)

Tabla 20

Escenario de calidad 2, disponibilidad.

Requerimiento no funcional	Disponibilidad	Prioridad	Alta
Justificación	El grupo de desarrollo establece la importancia de mantener disponibles los datos de los sensores para su consulta por parte del cliente		
Fuente	Cliente		
Estímulo	Al presionar el botón consultar de la aplicación móvil		
Artefacto	Aplicación móvil, componente web y dispositivo IoT		
Ambiente	Operación normal del componente web y los servicios de Thinker.io		
Respuesta	Ver la temperatura, humedad y pH de mi planta		
Medida de la respuesta	Disponibilidad de los datos en el 95% de las peticiones realizadas al sistema		

Nota. Fuente: Elaboración propia (2021)

Tabla 21

Escenario de calidad 3, mantenibilidad

Stakeholder	Mantenibilidad	Prioridad	Media
Justificación	El grupo de desarrollo establece la importancia de aplicar buenas prácticas de programación con el propósito de mantener un código fuente legible a cualquier desarrollador para hacer modificaciones a futuro.		
Fuente	Desarrollador.		
Estímulo	Modificación y versionamiento de nuevo código.		
Artefacto	Componente web, móvil.		
Ambiente	Proyecto configurado con repositorio Gitlab y Sonarcloud.		
Respuesta	Reporte de análisis estático en Sonarcloud.		
Medida de la respuesta	80% de cubrimiento de pruebas, 3% de líneas duplicadas, 0 vulnerabilidades, calificación A en vulnerabilidades		

Nota. Fuente: Elaboración propia (2021)

Resultado del análisis de los requerimientos funcionales y no funcionales se define el diseño y la arquitectura del sistema

7.3. Diseño y arquitectura

Para la definición de la arquitectura se tienen en cuenta características propias de las plantas y los requerimientos funcionales y no funcionales, las habilidades técnicas del equipo de trabajo, restricciones de recursos y de tiempo y una comparativa de los diferentes estilos arquitecturales.

Debido a que los valores de las variables a medir en las plantas como lo son humedad, temperatura y pH no presentan cambios drásticos inmediatos en el tiempo de forma natural, las

variaciones se dan de forma paulatina, por esta razón no se requiere una medición constante de las variables mencionadas

El equipo de trabajo si bien en cuenta con las capacidades técnicas a nivel de desarrollo de software y hardware, no se tiene experiencia en implementaciones de IoT donde, generalmente se implementan patrones arquitecturales de microservicios o eventos los cuales fueron comparados con otros patrones como se visualiza en la tabla 21 para determinar la arquitectura a utilizar:

Tabla 22

Comparativa patrones arquitectónicos

Arquitectura	**Agilidad**	**Despliegue**	**Testeabilidad**	**Rendimiento**	**Escalabilidad**	**desarrollo**
Capaz	1	1	3	1	1	3
Microservicios	3	3	2*	1	3	3**
Eventos	3	3	1	3	3	1
Microkernel	3	3***	3	3	3	3

Nota. La calificación es de 1 a 3 donde 1 es menor aplicación y 3 mayor aplicación de la característica de la fila 1. * depende de la descomposición de los servicios. ** En equipos pequeños que dominen los diferentes lenguajes que se pueden dar. *** Depende de la implementación principal donde se acoplan los plugins. Fuente: Elaboración propia.

Analizada la comparación y con respecto a las necesidades y prioridades del proyecto se considera la elección de arquitectura de capaz debido a que si bien

7.3.1. Diagrama de Despliegue

En la figura 23 se identifican 4 componente estructurales en una solución de IoT los cuales poseen las siguientes características:

Sensores

- **Humedad: YL-69:** Sensor con voltaje de operación de 5v, capaz de proyectar la humedad de terreno con una eficiencia suficiente para el ratio de lecturas requeridas por el sistema.
- **Temperatura y humedad ambiente: DHT11:** Se opta por sensor DHT11, voltaje de operación de 5V, capaz de proyectar de manera más precisa, la temperatura y humedad ambiente, el LM35 se recomienda para otros escenarios.
- **Board de Microprocesamiento: Arduino UNO:** Se opta por Arduino como sistema microprocesador, siendo un sistema que no requiere procesos embebidos ni complejos, es la más adecuada y versátil.
- **Gateway: Módulo ethernet W5500:** Se opta por un módulo ethernet, protocolos TCP/IP variados, para comunicación entre sistemas (hardware y software).

Gateway: Placa Arduino

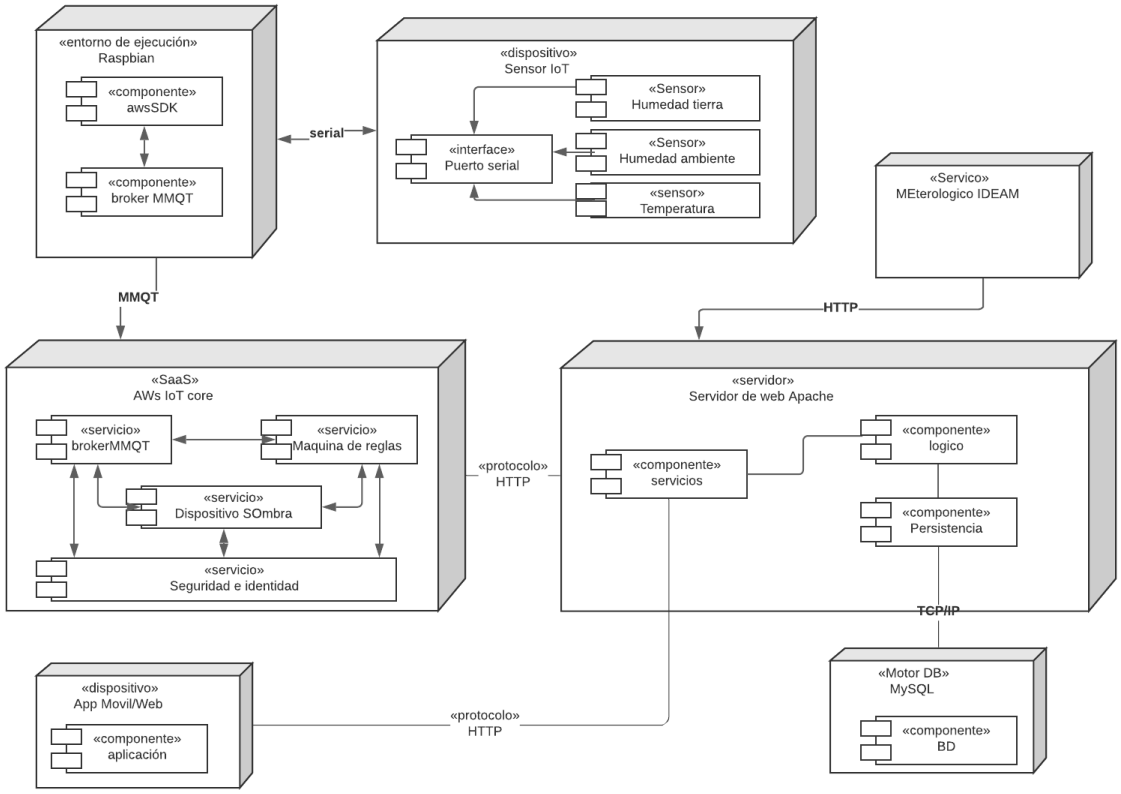
Nube IoT: THinker.io.

Servidor Web: Desplegado en un equipo con SO Ubuntu 18 server con servidor web Apache2 para soportar PHP versión 7.4 y Motor de base de Datos MySQl 5.7

App cliente: Soportado por dispositivos móviles iOS 8 y Android 7 (Nougat) en adelante.

Figura 23 Diagrama de Despliegue

Diagrama de Despliegue



Nota. Fuente: Elaboración propia (2021)

7.3.2. Caso de Uso arquitectura relevante.

Analizada la problemática y a resolver y de acuerdo con el objetivo general del proyecto se identifican los roles de la figura 24 con sus relaciones que permitieron identificar los requerimientos funcionales, arquitecturas viables y tecnologías a utilizar.

Figura 24

Caso de uso relevante

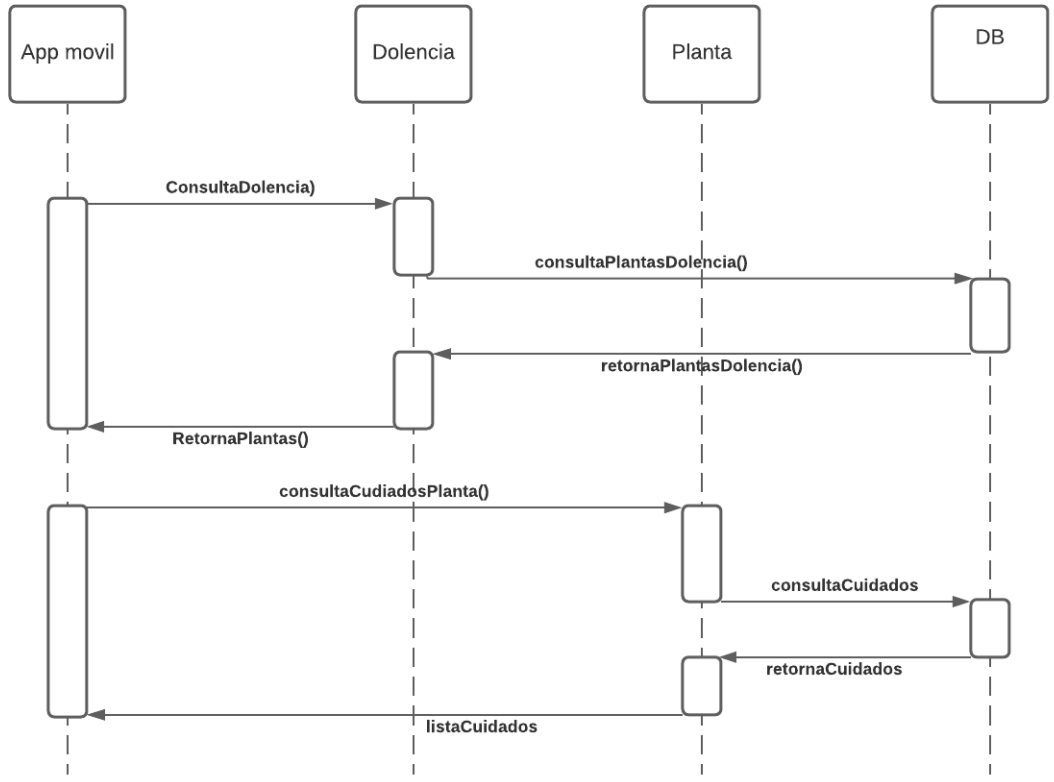
Nota. Fuente: Elaboración propia (2021)

7.3.3. Diagrama de secuencia

El diagrama de secuencia de la figura 25 se determinan la línea de vida del proceso a ejecutar en la funcionalidad consultar plantas a partir de una dolencia determinada seleccionada por el usuario.

Figura 25 *Diagrama de secuencia, consulta de plantas.*

Diagrama de secuencia, consulta de plantas.

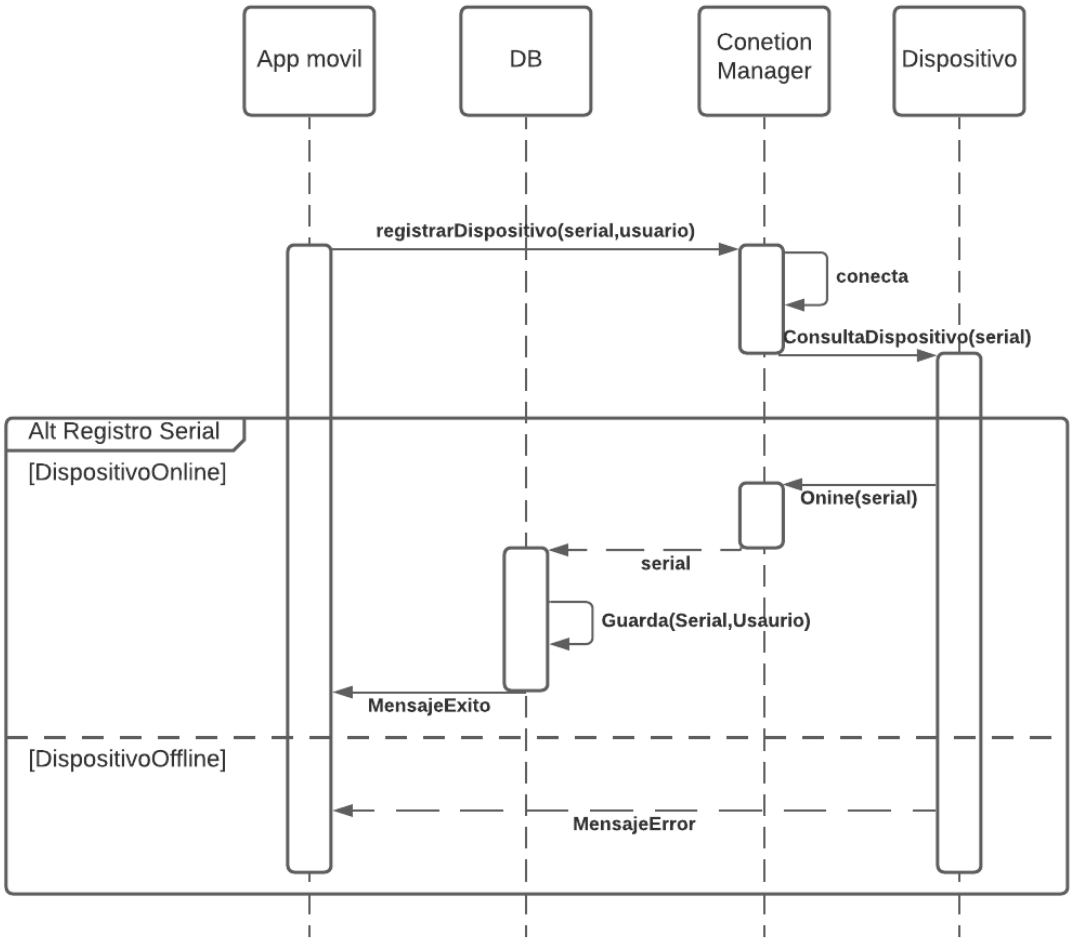


Nota. Fuente: Elaboración propia (2021)

En cuanto al proceso de activación de un dispositivo IoT en el sistema el diagrama de secuencia que se puede aprecia en la figura 26 se determina el flujo de proceso para identificar si el dispositivo a registrar hace parte del sistema previo registro en el servicio AWS IoT por parte del equipo de desarrollo.

Figura 26

Diagrama de secuencia Registro de Dispositivo IoT en el sistema

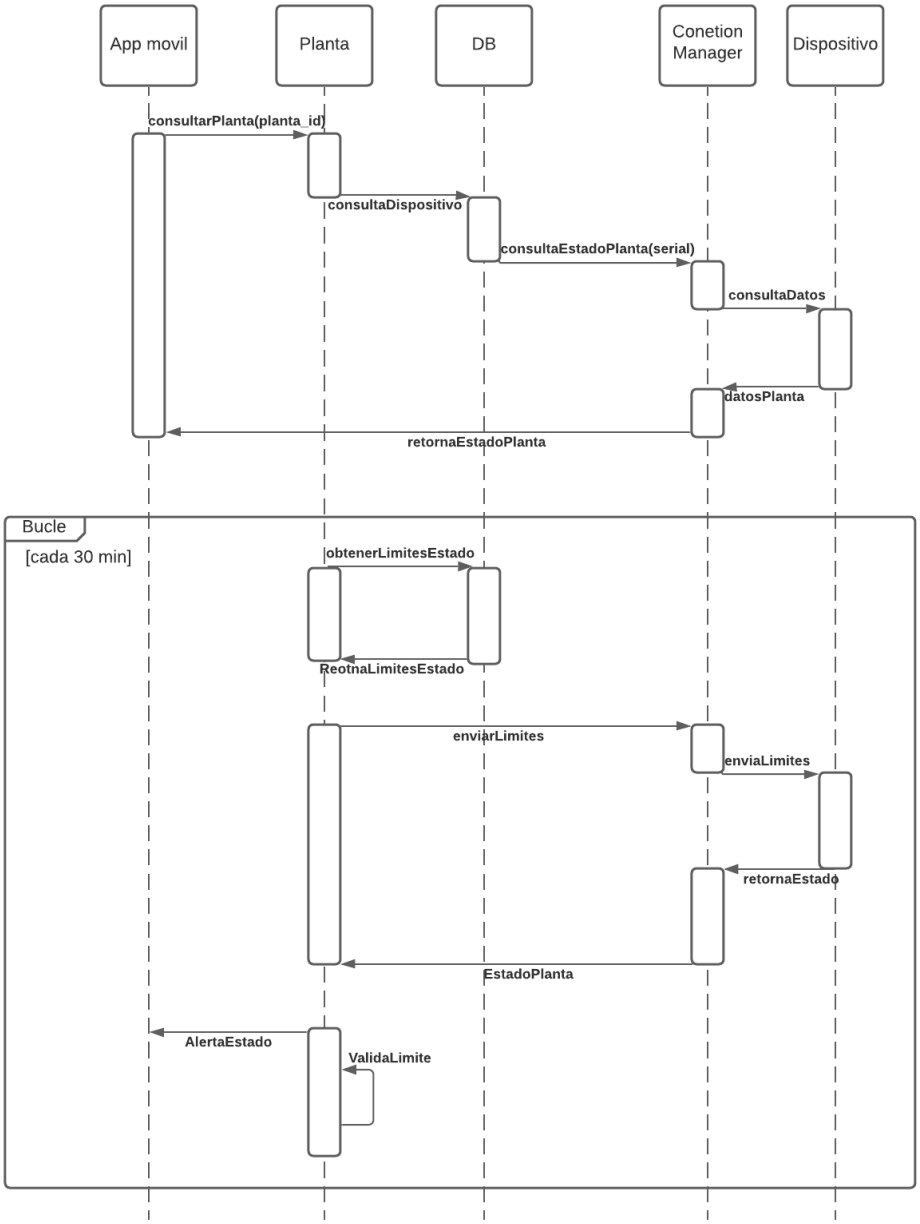


Nota. Fuente: Elaboración propia (2021)

En la figura 27 se aprecia el proceso a ejecutar en para consultar las condiciones de la planta a petición del usuario, adicionalmente se visualiza el ciclo de monitoreo de la planta para detectar cuando supera los límites de temperatura, humedad y pH parametrizados en el sistema:

Figura 27

Diagrama de secuencia, Consulta estado planta y función de alertas.



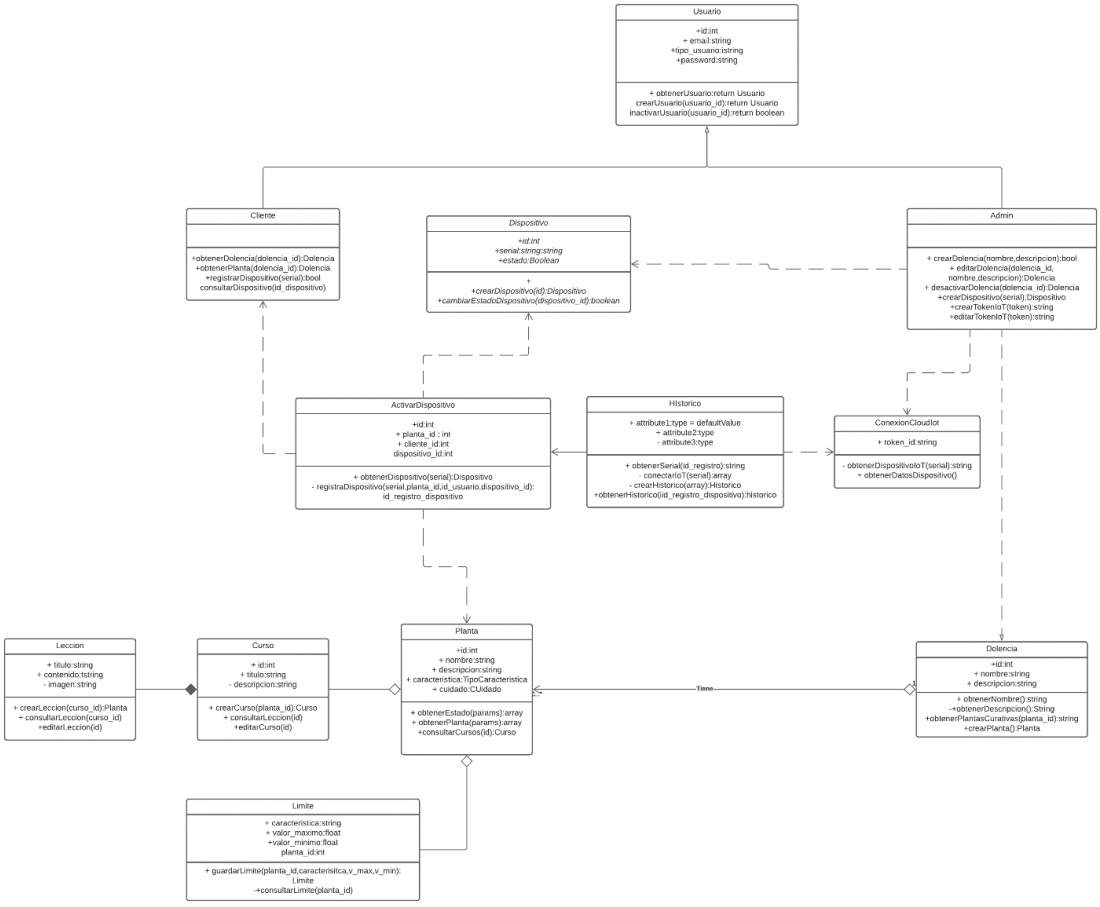
Nota. Fuente: Elaboración propia (2021)

7.3.4. Diagrama de clases

La estructura del sistema se define con el diagrama de clases de la figura 28, bajo un enfoque de Programación Orientada a Objetos en este diagrama se determina las clases, relaciones y funciones a partir de la abstracción del problema a resolver con sistema a desarrollar identificando entidades como planta, dolencia, usuarios entre otros.

Figura 28

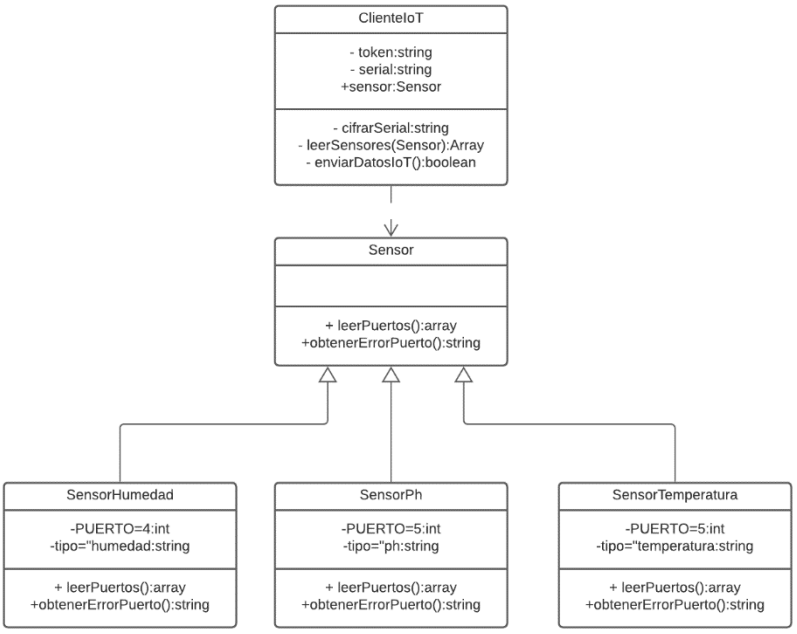
Diagrama de clases



Nota. Fuente: Elaboración propia (2021)

Figura 29

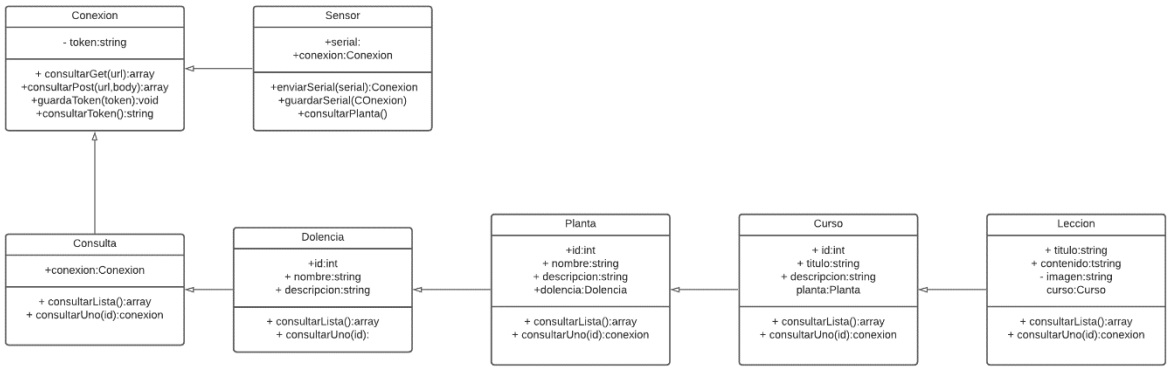
Diagrama de clases gateway



Nota. Fuente: Elaboración propia (2021)

Figura 30

Diagrama de Clases aplicación móvil



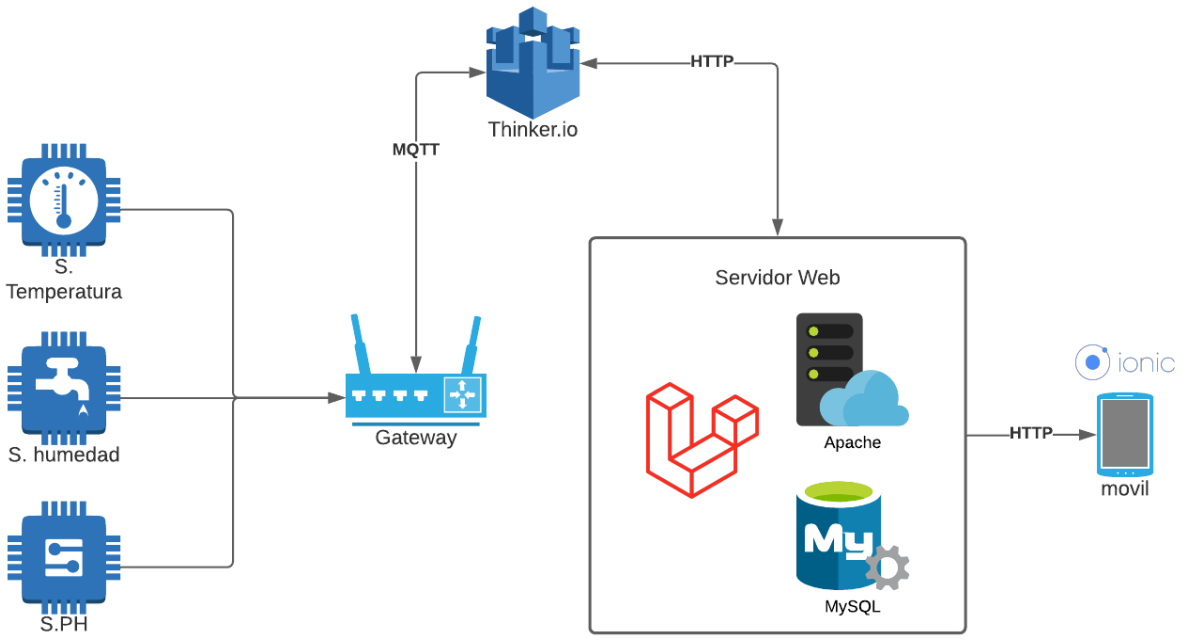
Nota. Fuente: Elaboración propia (2021)

7.3.5. Arquitectura de alto nivel

Con la arquitectura de alto nivel de la figura 29 se define la estructura general del sistema donde se identifican componentes clave para el funcionamiento, este diseño se implementó de acuerdo con las arquitecturas tradicionales de soluciones de IoT del mercado donde se definen los sensores a utilizar, la nube IoT, el servidor web del sistema y un cliente móvil.

Figura 31

Arquitectura de alto nivel



Nota. Fuente: Elaboración propia (2021)

7.4. Construcción

Para abordar la construcción del sistema se realizó inicialmente una investigación de tecnologías y proveedores de IoT en la nube, cuyo resumen se puede observar en la tabla 24, para delegar aspectos de seguridad y de administración de los dispositivos en los dos aspectos claramente diferenciado de software y hardware, como se menciona en el apartado de metodología.

Tabla 23 Diferencias de servicios de IoT en la nube

Diferencias de servicios de IoT en la nube

	AWS IoT	MS Azure IoT	Thinker Io
Licencia	Propietario	Propietario	OpenSource
protocolos	Control: HTTPS. Datos: HTTPS, WebSockets y MQTT	HTTP, AMQP, MQTT y protocolos personalizados.	MQTT, CoAP, HTTP, PSON
Comunicación	Telemetría (Control) y Mando	Telemetría (Control) y Mando	
Dispositivos soportados	Broadcom, Marvell, Renesas, Texas Instruments, Microchip, Intel, Mediatek, Qualcomm, Seeed, BeagleBoard, Raspberry Pi, Inte	Intel, Raspberry Pi, Freescale, Texas Instruments, MinnowBoard, BeagleBoard, Seeed, resin.io	Arduino, Linux, Raspberry Pi, MQTT clientes
SDK	Embedded C, JavaScript, Arduino Yún	.Net y UWP, Java, C, Nodejs	N/A
Seguridad	TLS (Mutua autenticación) T	TLS (Solo autenticación de servidor)	TLS (Solo autenticación de servidor)
Autenticación	AWS IAM o AWS Cognito para solicitudes HTTPS y WebSockets. SigV4 y Certificado X.509 para conexiones con HTTP y autenticación basado en certificados para conexiones MQTT	Por dispositivo con el token de SAS	Por dispositivo con el token de SAS

Nota. Fuente: Elaboración propia (2021)

Una vez analizadas las opciones se decide por Tinker.io debido a que posee las características de comunicación necesarias para el proyecto y, dado que no se van a manipular datos sensibles, se considera que o se requieren protocolos de seguridad tan fuertes.

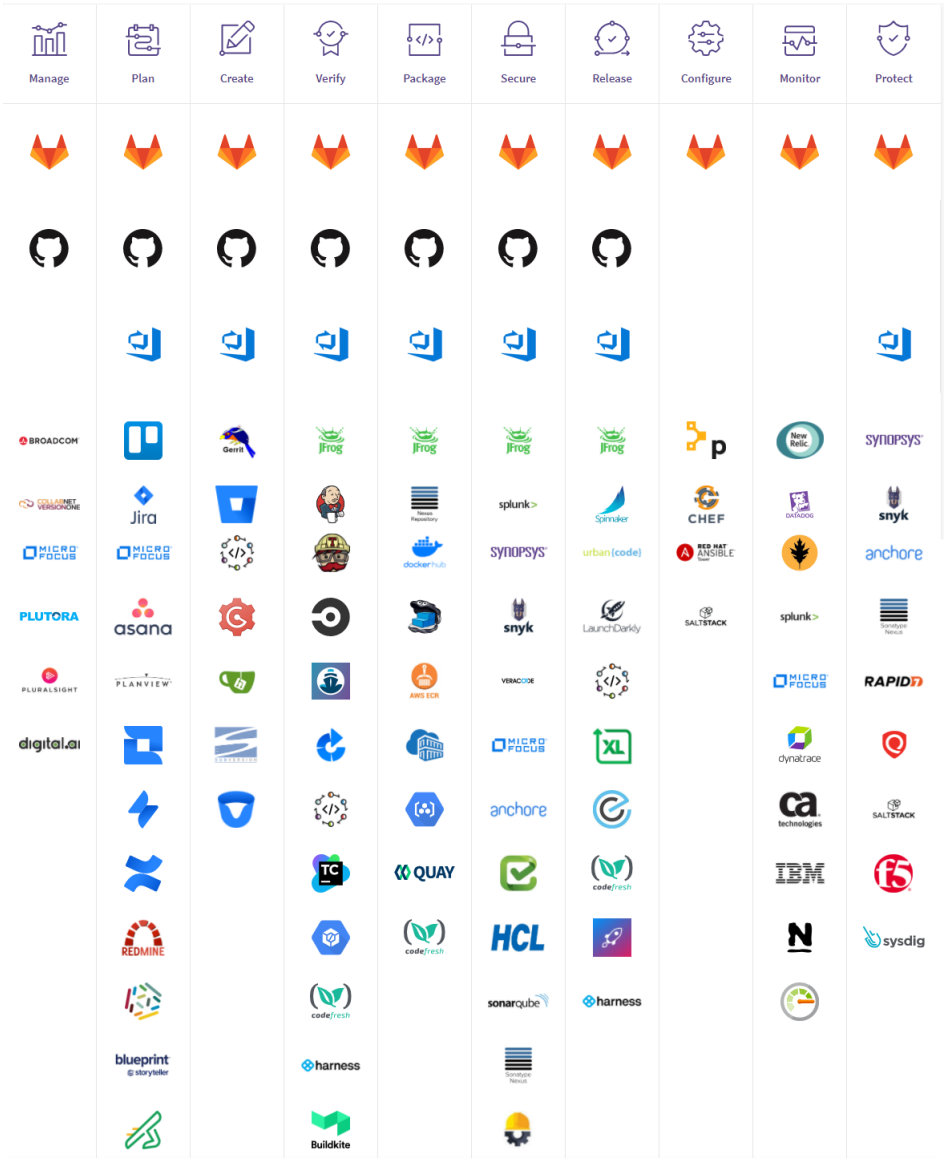
Posteriormente, identificadas las habilidades duras del equipo se definen las tareas a partir del campo de acción aplicables al hardware y software donde Wilson Contreras asume el desarrollo e implementación al sistema de los componentes de hardware y David Jiménez asume el desarrollo web y móvil del sistema. Lo anterior permite un mejor trabajo en paralelo.

A fin de mantener una copia de seguridad, realizar rollbacks y procedimientos de integración continua se consideran diferentes herramientas para llevar un mejor control del

proyecto relacionado con la construcción. En la figura 30 se aprecia la comparativa de funcionalidades de diferentes productos en el mercado.

Figura 32 Comparativa soluciones DevOps

Comparativa soluciones DevOps



Nota. Fuente: Gitlab.com

En cuanto al software se procede a realizar los diseños bajo el Lenguaje Unificado de Modelado UML cuyos resultados se pueden apreciar en el apartado de Diseño y Arquitectura.

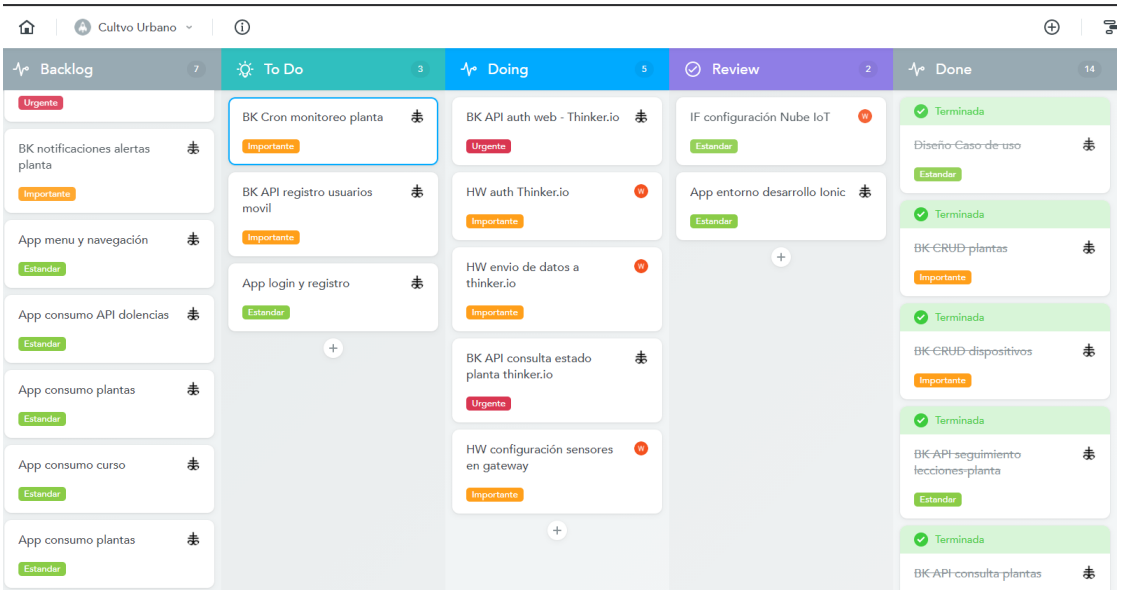
A fin de agilizar el desarrollo se escoge el framework de PHP Laravel que tiene funciones prestablecidas y listas para la implementación como lo son autentificación, roles, protección de rutas web y API, entre otras. Adicionalmente se presta para implementar un modelo de capaz donde se identifican las diferentes responsabilidades de los artefactos de software.

Respondiendo a la arquitectura de alto nivel que se puede apreciar en la figura 29, para el desarrollo de la aplicación móvil se considera el uso de Ionic framework dado que el equipo no cuenta con experiencia en el desarrollo nativo de aplicaciones Android y iOS y dado que la aplicación no requiere de contenidos multimedia muy elaborados se considera que una tecnología hibrida basada en web permite una curva de aprendizaje más rápida

Debido a los límites de trabajo en proceso en un estado del tablero Kanban definido en el apartado de metodología se identificaron en el desarrollo del proyecto inconvenientes relacionados con la conectividad con la nube de IoT seleccionada. Como se visualiza en la figura 31, la columna Doing llega a limite

Figura 33

Límite de trabajo alcanzado



Nota. Fuente: Elaboración propia (2021)

Adicionalmente, con el esquema de tipificación de tareas relacionado en el apartado de metodología, si bien la construcción tiene enfoques diferentes comparten aspectos comunes como la seguridad, la calidad y buenas prácticas en el desarrollo, por lo anterior una vez verificado los requerimientos funcionales, no funcionales y realizados los diseños relacionados en sus respectivos apartados del presente documento se abarca la construcción en dos frentes, hardware y software.

En cuanto a la construcción de software, durante el proceso de implementación de los diseños relacionados en el apartado Diseño y Arquitectura se tuvieron en cuenta dos aspectos fundamentales en cuenta, calidad y seguridad.

En cuanto a calidad, el uso del framework Laravel permite la implementación de buenas prácticas tales como el empaquetamiento de artefactos de acuerdo con su función, a continuación, se describen 4 grupos principales:

Los modelos, agrupados en la carpeta model y se encargan del acceso a datos.

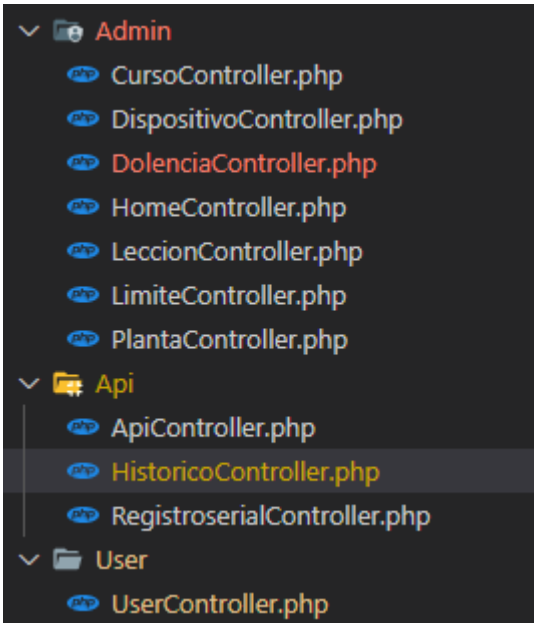
Los controladores, agrupados en la carpeta controllers y contiene la lógica de negocio.

Las vistas, que aplican en el administrador web se agrupan en la carpeta view

Adicionalmente se realizan unas subcategorías en los controladores acordes al proyecto como se visualiza en la figura 33

Figura 34 *Definición de paquetes*

Ejemplo de definición de paquetes



Nota. Fuente: Elaboración propia (2021)

También se define de forma las diferentes clases de acuerdo con su responsabilidad, donde, por ejemplo, Dispositivo cuenta con todas las funciones relacionadas con este como el registro de seriales, consulta de datos al dispositivo de IoT, desactivación de dispositivos entre otras.

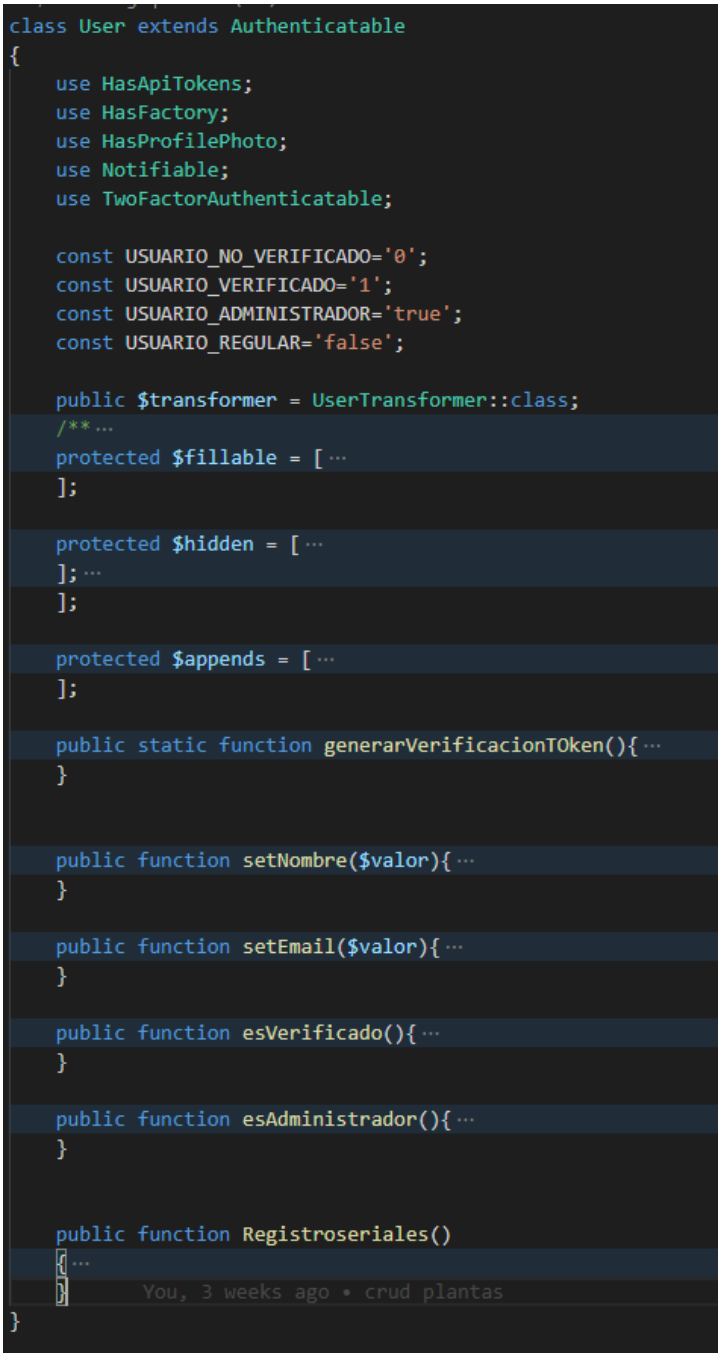
Adicionalmente, sobre la codificación de clases se define la siguiente nomenclatura

El nombre de la Clase aplica la nomenclatura Upper CamelCase, con la primera letra de cada palabra en mayúscula,

Para atributos se registran primero las constantes en mayúscula toda la palabra y con separador ‘_’ en caso de usar más de dos palabras a fin de tener nombres de atributos descriptivos, posteriormente se definen atributos protegidos y luego públicos en minúscula, al igual que las constantes se maneja el ‘_’ como separado de palabras. Los métodos se organizan en el orden de privados, públicos estáticos y públicos aplicando Camell Case en su definición como se ve en la figura 35.

Figura 35

Ejemplo definición estructura de clases



```
class User extends Authenticatable
{
    use HasApiTokens;
    use HasFactory;
    use HasProfilePhoto;
    use Notifiable;
    use TwoFactorAuthenticatable;

    const USUARIO_NO_VERIFICADO='0';
    const USUARIO_VERIFICADO='1';
    const USUARIO_ADMINISTRADOR='true';
    const USUARIO_REGULAR='false';

    public $transformer = UserTransformer::class;
    /** ...
    protected $fillable = [ ...
    ];

    protected $hidden = [ ...
    ]; ...
    ];

    protected $appends = [ ...
    ];

    public static function generarVerificacionTOken(){ ...
    }


    public function setNombre($valor){ ...
    }

    public function setEmail($valor){ ...
    }

    public function esVerificado(){ ...
    }

    public function esAdministrador(){ ...
    }


    public function Registroseriales()
    { ...
    }       You, 3 weeks ago • crud plantas
}
```

Nota. Fuente: Elaboración propia (2021)

El aspecto de seguridad se realiza un análisis de riesgo cuyo detalle se puede consultar en el anexo A, este permitió identificar algunos riesgos sobre los cuales realizar los requerimientos de seguridad, insumo fundamental para la adopción del SDL (Security Development Lifecycle) donde se definen el entrenamiento, requerimientos de seguridad, su diseño, implementación y comprobación como se puede observar en las tablas 24 a 28.

Tabla 24

Contenido de la etapa de entrenamiento definido para el SDL

Componente	Entrenamiento
Dispositivo IoT	OWASP IoT Top 10.
	OWASP Firmware Security Testing Methodology.
Api	Owasp Api Security.
Web	OWASP Top Ten.
Móvil	Owasp MSTG (Mobile Security testing guide.
	Owasp MASVS (Mobile Application Security Verification Standard).

Nota. Fuente: Elaboración propia (2021)

Tabla 25

Etapa de requerimientos de seguridad definido para SDL

ID	Requisito	Descripción	Criterio de aceptación
Aplicación móvil			
R1	Servicios de red seguros.	A fin de brindar confianza y evitar ataques informáticos.	Consumo endpoints con certificado SSL instalado.
R2	Autenticación y autorización	Función de registro, login y autorización de peticiones.	Usuarios registrados en el sistema realizan consultas al dispositivo sensor.
App web			
R3	Exponer en el api solo la información requerida por el usuario.	No exponer campos adicionales a fin de ocultar en lo posible la estructura de la DB.	No se deben exponer ningún dato adicional al necesario.
R4	Prevenir ataques de falsificación de petición en sitios cruzados.	Se debe identificar peticiones validas del administrador al sistema.	Token en cada uno de los formularios del administrador.

R5	Mecanismo de autenticación en el consumo de la API.	EL servicio solo debe responder a usuarios registrados	Todas las peticiones deben asociar un token de seguridad previo login del usuario.
R6	Controles SQL para evitar la fuga masiva de datos en caso de inyecciones SQL	Par evitar ataque donde a través de la manipulación de las peticiones obre la API se puede agregar sentencias maliciosas SQL.	Todas las consultas a DB se deben realizar a través del ORM.
Dispositivo IoT			
R7	Seguridad en Hardware, evitando manipulación de la board por parte de un tercero.	A fin de evitar el envío de datos errados	Detección del dispositivo manipulado para su desactivación por el administrador.
R8	Impedir que dispositivos falsificados se conecten al sistema	Solo los dispositivos configurados por el administrador pueden conectarse al sistema	Detección de dichos dispositivos para su anulación por parte del administrador

Nota. Fuente: Elaboración propia (2021)

Tabla 26

Etapa de diseño definido para SDL

ID	Requisito	Diseño
Aplicación móvil		
R1	Servicios de red seguros.	Certificado SSL.
R2	Autenticación y autorización.	Autenticación OAuth2: Implementación de Servidor PHP OAuth 2.0. De acuerdo con las especificaciones RFC6749 de la IETF La aplicación cliente envía su id y código de autorización parametrizado en el sistema. Recibe un código de autorización. Dicho código es enviado junto con el usuario y contraseña de un usuario registrado De ser exitoso el servidor responde un token de acceso personal que se valida en todas las peticiones que se hagan al sistema.
App web		
R3	Exponer en la api solo la información requerida por el usuario.	Restricción de campos expuestos. Se definen los datos necesarios para su consumo de cara a la aplicación móvil(títulos, descripciones, imágenes, etc.), los demás se dejan como atributos ocultos en las clases definidas en el ORM que acceden a los datos. Redefinición de campos. Construcción una capa de presentación y transformación de los nombres de campos con la librería Fractal, se mapea los campos expuestos y se asigna correspondiente alias, de esta manera no se exponen los nombres de los campos de la base de datos, tales como id, name, cliente_id, entre otros.
R4	Prevenir ataques de falsificación de petición en sitios cruzados	Generación token CSRF para cada sesión de usuario.
R5	Mecanismo de autenticación en el consumo de la API	Autenticación OAuth2.

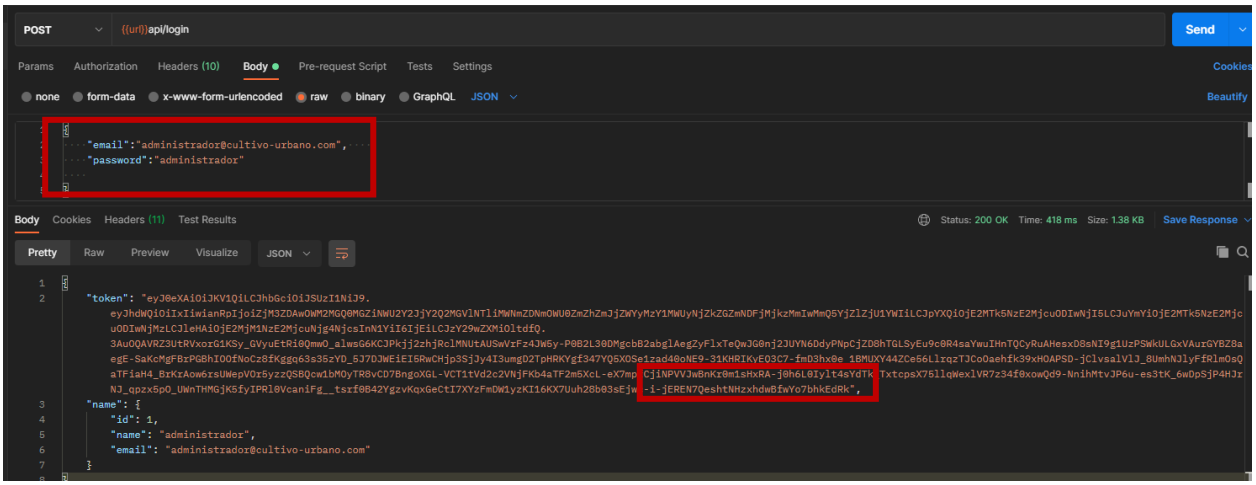
R6	Controles SQL para evitar la fuga masiva de datos en caso de inyecciones SQL	ORM para el acceso a DB. Se implementa Elloquent ORM de la versión 8.0 del framewrok Laravel. Los datos de las tablas se mapean a objetos. Para la realización de transacciones se aplica el patrón Active Record, las operaciones de la base de datos son realizadas por el ORM para no utilizar sentencias SQL en la capa lógica.
Dispositivo IoT		
R7	Seguridad en Hardware, evitando manipulación de la board por parte de un tercero	Circuito de protección, modifica la bandera a falso si el case del dispositivo es abierto.
R8	Impedir que dispositivos falsificados se conecten al sistema	Registro de seriales encriptados en el dispositivo para su identificación en la nube IoT y el administrador web del sistema mediante OpenSSL y el cifrado AES-256. Todos los valores cifrados estarán firmados con un código de autenticación de mensajes (MAC) con el cual se impide el descifrado de seriales manipulados.

Nota. Fuente: Elaboración propia (2021)

Un de los aspectos que se consideró importante dentro del sistema es la autenticación y autorización, para lo cual, como se menciona en la tabla 26 se implementa la especificación OAuth2 cuyo funcionamiento se puede observar en las imágenes 36 y 37.

Figura 36 *Login y obtención de autorización con Auth2*

Login y obtención de autorización con Auth2

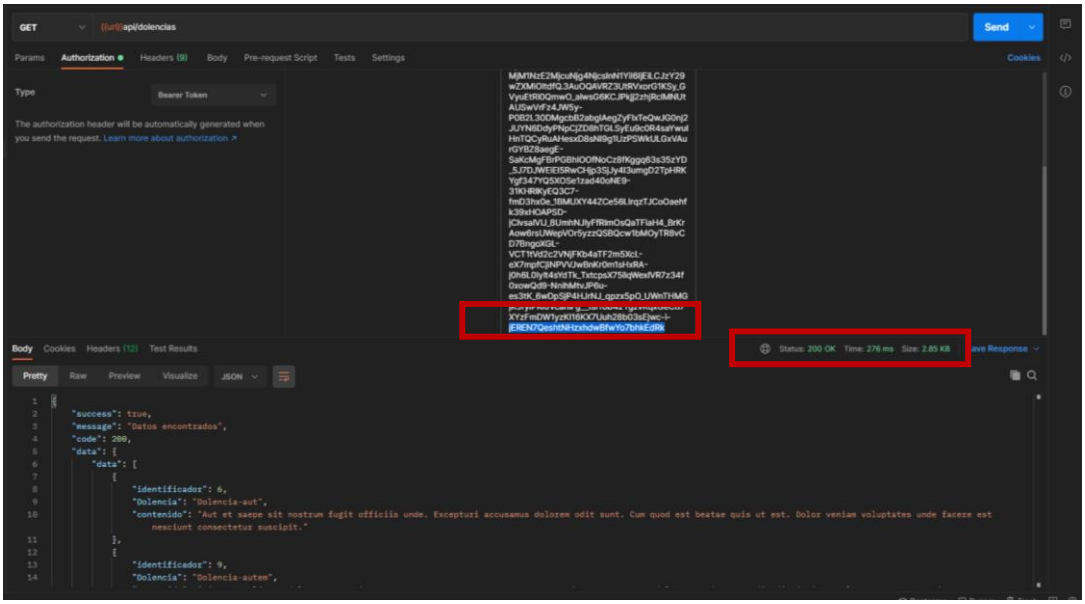


Nota. Fuente: Elaboración propia (2021)

En la imagen 37 se aprecia una petición exitosa enviando el token de autorización obtenido

Figura 37

Solicitud de recurso con token de autorización



Para la etapa de implementación se realiza un inventario de las herramientas a utilizar para verificar vulnerabilidades conocidas sobre estas se utiliza la base de datos de https://cve.mitre.org/[1]

Tabla 27

Etapa de implementación definido para SDL

Herramienta	Librerías de terceros	Observaciones
App		
Ionic V.6	cordova-plugin-advanced-http	Sin vulnerabilidad conocida.
Administrador Web		
Php 7.4	No aplica	Se encuentran vulnerabilidades relacionadas proyectos realizados con php 7.4 como OS4Ed y YouPHPTUbe 7.4.
Laravel Framework 8	Passport Eloquent ORM Fractal Servidor OAtuh2	Vulnerabilidades conocidas para versiones anteriores a 8.4.2 en la configuración por defecto debug.
MySQL	No aplica	Se encuentran vulnerabilidades relacionadas con plugins, implementaciones y entornos que no se utilizarán en el proyecto como SO windows, uso de versiones de php anteriores a la 7.4 entre otros.
Dispositivo IoT		
Python 3.7	dht11 0.1.0 RPi.GPIO 07.0 paho-mqtt 1.5.1	Sin vulnerabilidad conocida
Proveedor IoT Cloud		
Ubidots.com	Servicio MQTT broker	Sin vulnerabilidad conocida

Nota. Fuente: Elaboración propia (2021)

[1] CVE (Common Vulnerabilities and Exposures), es un registro de vulnerabilidades de seguridad conocidas mantenido y supervisado por la corporación MITRE con recursos de la Agencia de Seguridad de Infraestructura y Ciberseguridad de Estados Unidos.

Adicionalmente, en la etapa de implementación se realiza un análisis estático en la herramienta Sonar Cloud, cuyos resultados se pueden observar en las figuras 39 y 40 del apartado Pruebas.

Dados los resultados obtenidos y el alcance del prototipo a realizar en el proyecto se considera que las herramientas escogidas para el desarrollo de este pueden satisfacer los objetivos propuestos.

En la tabla 28 se especifica las herramientas y componentes a analizar cuyos resultados se pueden apreciar en las figuras 53 a 72 del apartado Pruebas del presente documento

Tabla 28

Definición de pruebas etapa de comprobación.

Herramienta	**Componente**	**Objetivo**
Escaneo de puertos		
nmap	Cultivo Urbano	http://cultivo-urbano.dagodigital.com/
nmap	Proveedor IoT cloud	https://industrial.api.ubidots.com/
Captura de trafico		
whireshark	Proveedor IoT cloud	https://industrial.api.ubidots.com/
Análisis estático		
SonarCLoud	Cultivo Urbano	http://cultivo-urbano.dagodigital.com/
Análisis dinámico		
OWASP ZAP	Cultivo Urbano	http://cultivo-urbano.dagodigital.com/
OWASP ZAP	Proveedor IoT cloud	https://industrial.api.ubidots.com/

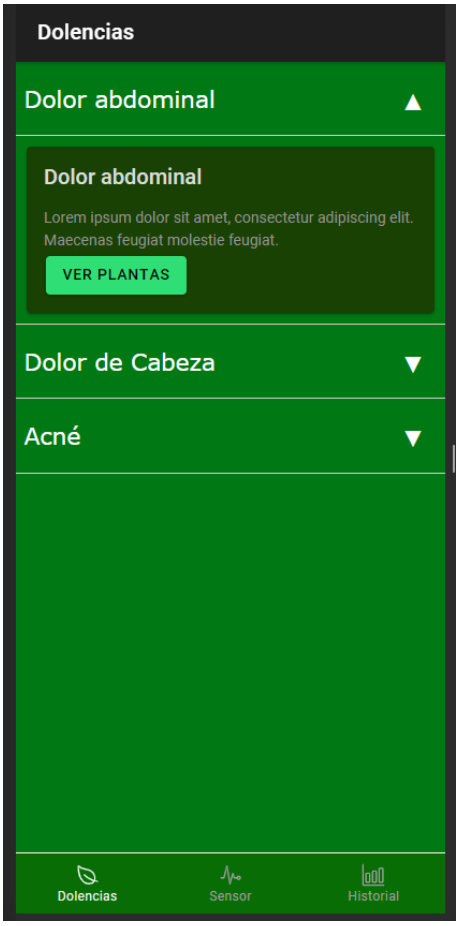
Nota. Fuente: Elaboración propia (2021)

La implementación del prototipo funciona de la aplicación móvil desarrollado con el framework Ionic responde al diagrama de clases de la figura 30 del apartado Diseño y Arquitectura. En cuanto a funcionalidad se realiza el diseño de la aplicación a partir de la identificación de los requerimientos funcionales donde se implementa un menú de tipo Tabs que

ofrece el framework a fin de distinguir los dos enfoques de la aplicación, el pedagógico, y el sensor.

Figura 38

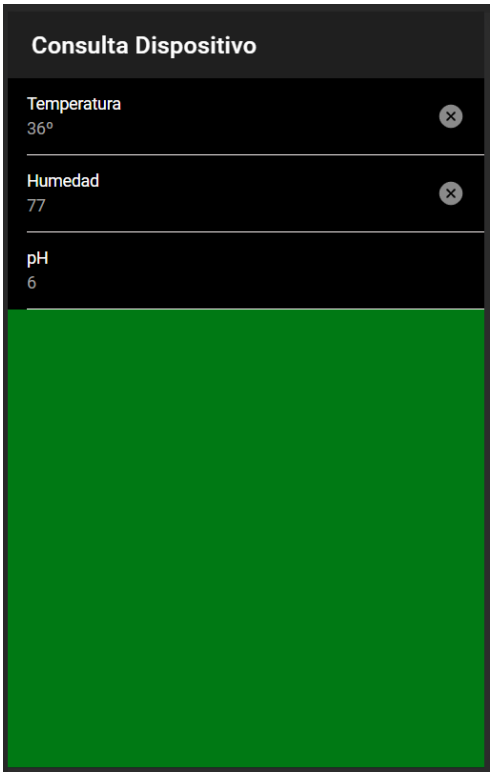
Menú consulta de dolencias y plantas



Nota. Fuente: Elaboración propia (2021)

Figura 39 Menú consulta sensor

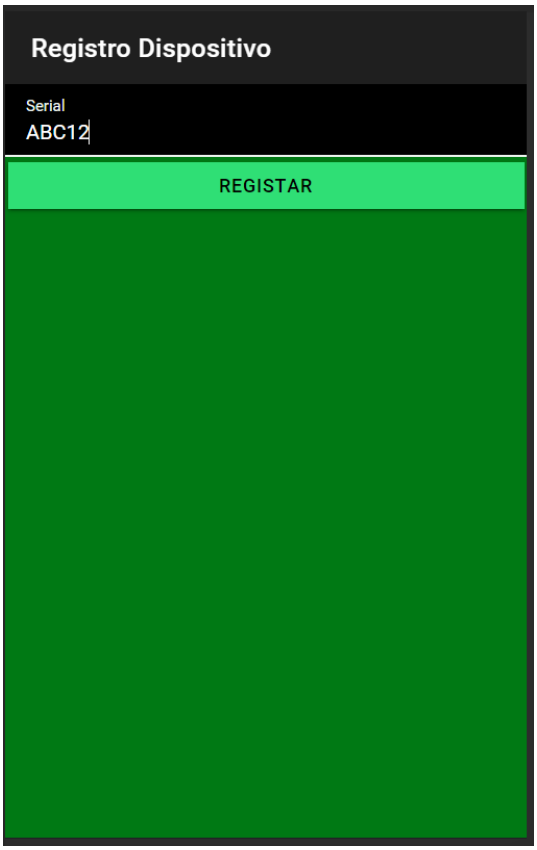
Menú consulta sensor



Nota. Fuente: Elaboración propia (2021)

Figura 40

Menú consulta sensor



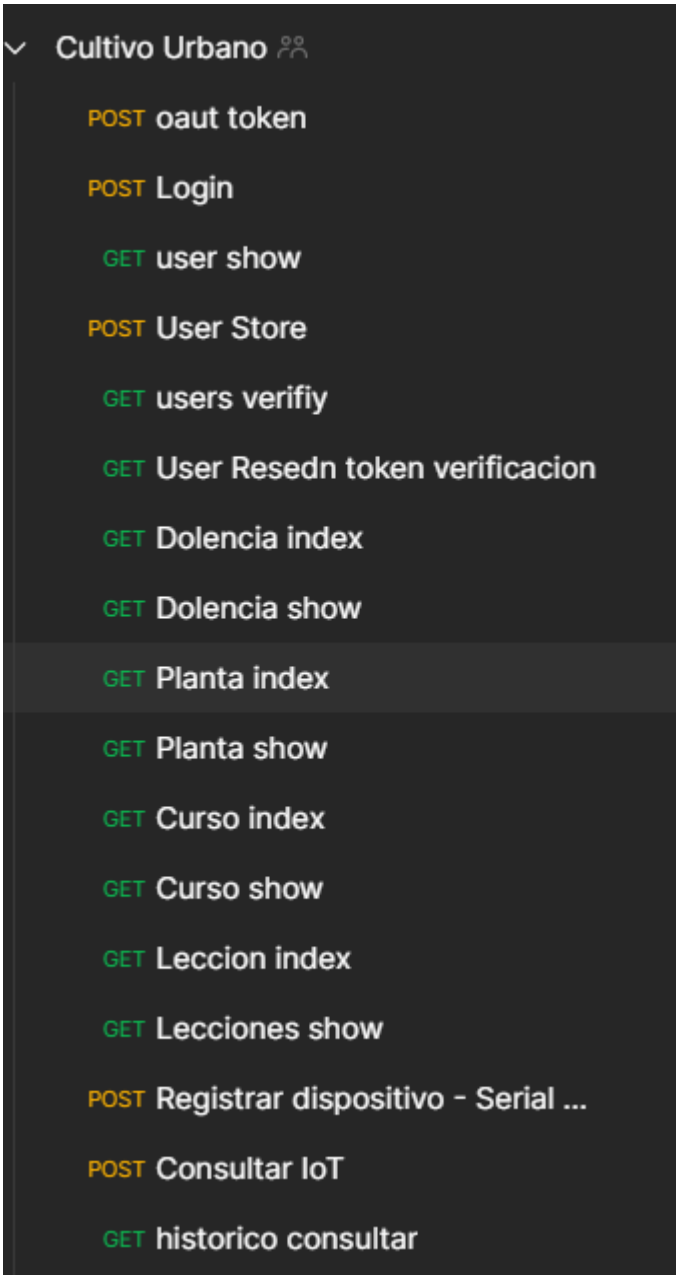
Nota. Fuente: Elaboración propia (2021)

Dado que el alcance del producto es un prototipo se hizo más énfasis en la usabilidad dando menos prioridad al diseño dado que no se cuenta con el recurso humano para la definición de un estilo gráfico.

Otro factor que se tiene en cuenta es el principio de responsabilidad única extrapolado al diseño donde cada interacción del usuario realiza el consumo de servicios expuesto por el componente web, en la figura 41 se relacionan los endpoint expuestos para la aplicación.

Figura 41

Endpoints expuestos por el componente web



Nota. Fuente: Elaboración propia (2021)

En cuanto al Hardware, se lleva a cabo la caracterización y linealización correspondiente para el sensor DHT11, ya que, los sensores no arrojan valores de temperatura finales. En principio se procede con definir los alcances y características propias del sensor.

Tabla 29 Datos de caracterización DHT11

Datos de caracterización DHT11

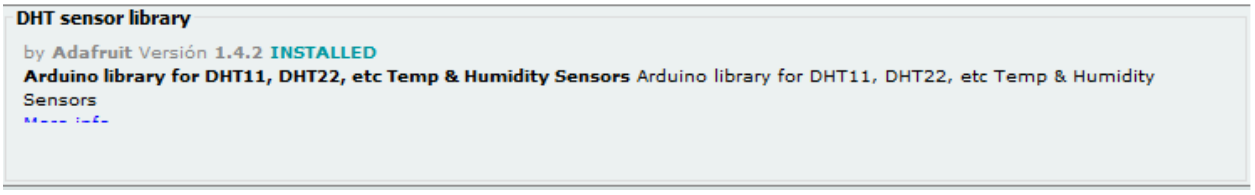
Característica	**Descripción**
Circuito integrado	Sí
Rango temperatura	0° a 50°
Precisión temperatura	1°C aprox
Rango de humedad	0% a 100%
Precisión de humedad	5% aprox
Resolución de muestreo análogo	0 - 1023
Frecuencia máxima de muestreo registrada	1 muestra/s

Nota. Fuente: Elaboración propia (2021)

Según los datos registrados conforme a la resolución y los valores mínimos/máximos, se procede a definir la conversión de valores para delimitar el muestreo de 0° a 50° para temperatura y de 0% a 100% para humedad. Para lo cual se hace uso de la librería propia del sensor DHT para Arduino.

Figura 42

Librería Arduino para sensor DHT11

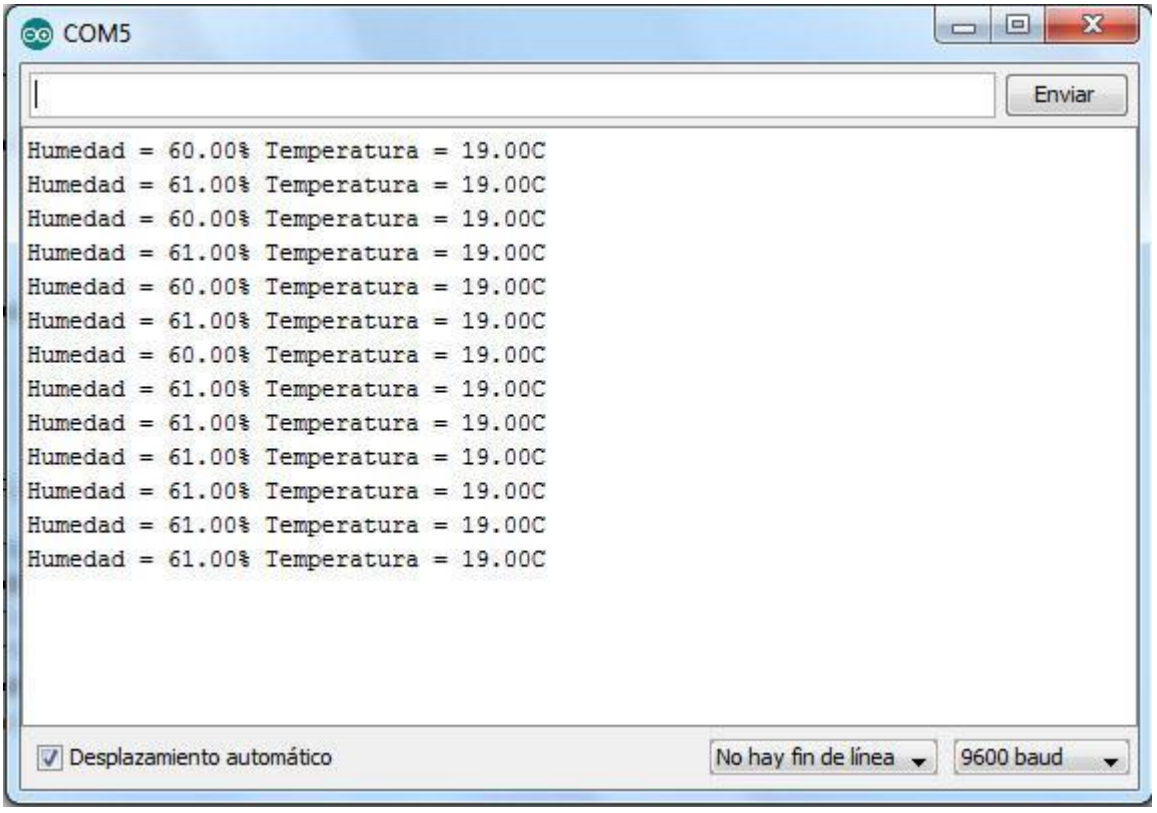


Nota. Fuente: Elaboración propia (2021)

Definida la librería, se procede a comprobar el funcionamiento correcto del sensor. El cual se aprecia en la figura 43.

Figura 43

Monitor serial de registro de DHT11

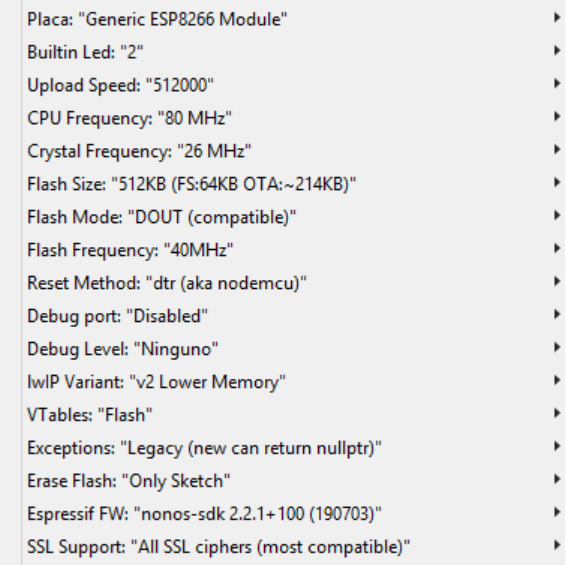


Nota. Fuente: Elaboración propia (2021)

Corroborando el funcionamiento del sensor con la librería incluida, se procede a incluir también, la librería del integrado ESP8266, el cual se encargará de cargar el código de trabajo y enviarlo vía Wifi. La librería cuenta también con la definición estándar de valores del módulo como la velocidad de carga, la variante IwIP del dispositivo, el reloj de la CPU del microprocesado, entre otros. Todos los valores se configurarán de la siguiente manera.

Figura 44

Configuración de sensor ESP8266

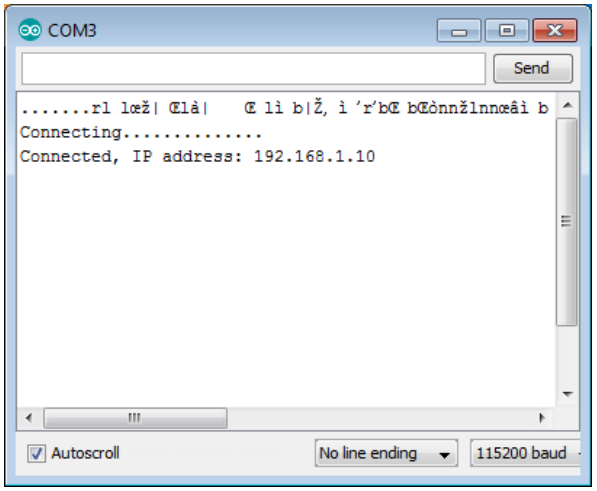


Nota. Fuente: Elaboración propia (2021)

Se prueba el funcionamiento del módulo, conectándolo a una red wifi valida, si el dispositivo, su configuración y las librerías son correctas, el sistema enviará un mensaje de confirmación de conexión, como se aprecia en la figura 45

Figura 45 Monitor serial de registro de ESP8266

Monitor serial de registro de ESP8266

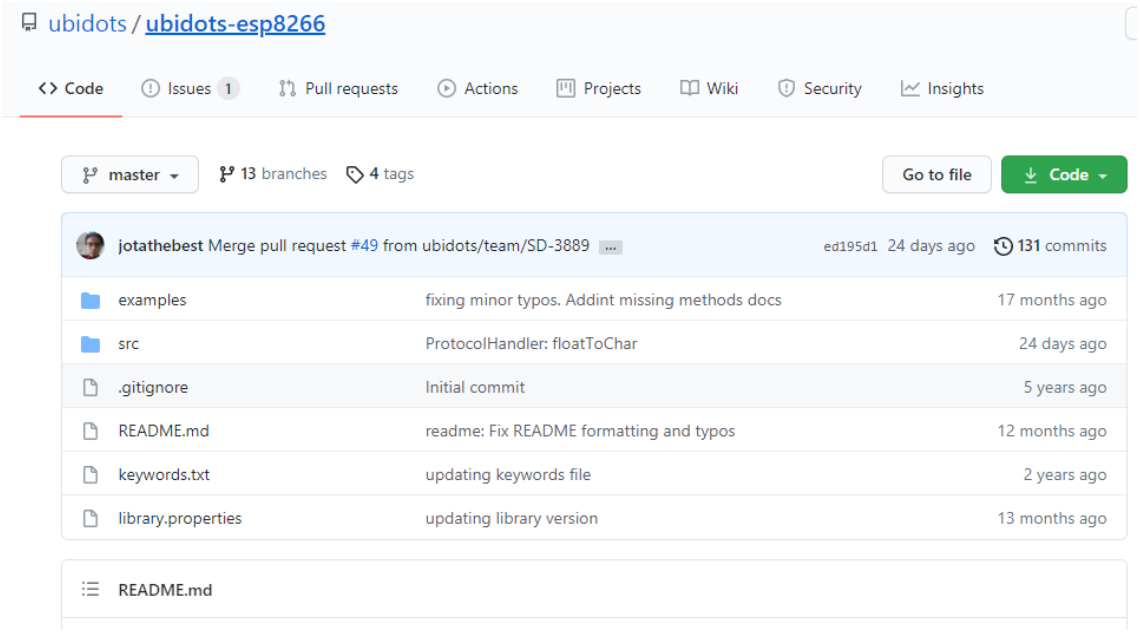


Nota. Fuente: Elaboración propia (2021)

Con el objetivo de registrar los datos en la nube, se hace uso de una plataforma IoT conocida como “Ubidots”, esta plataforma también requiere sus respectivas librerías y deben ser descargadas directamente desde el repositorio GitHub de Arduino.

Figura 46

Repositorio Github para librerías de Ubidots Arduino.



Nota. Fuente: Elaboración propia (2021)

Se valida que los módulos funcionan correctamente, por lo que se procede a llevar a cabo la implementación de sensores y módulos de procesamiento. Se definen primero las librerías y variables con las que el programa va a soportarse. Figura 42.

Figura 47

Variables y librerías del código de procesamiento.

```
#include "Ubidots.h"
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN 4
#define DEVICE  "ESP8266_Project"  // Nombres de variables de Ubidots
#define ID_TEMP "temperatura"
#define ID_HUM "humedad"
#define ID_PH "ph"
#define TOKEN  "BBFF-mXS8oAYv72nDQrYn8qvSFGf8LeLgRc"  // Token de Ubidots

#define WIFISSID [redacted] // Nombre WiFi
#define PASSWORD [redacted] // Password WiFi

#define DHTTYPE    DHT11
DHT_Unified dht(DHTPIN, DHTTYPE);

uint32_t delayMS;
Ubidots client(TOKEN);
float Valor_Temperatura = 0;
```

Nota. Fuente: Elaboración propia (2021)

Posterior a eso, se define el SetUp del código, con comandos de inicio de variables y periferias del sistema.

Figura 48

Setup del código de procesamiento

```
void setup() {
    Serial.begin(115200);

    client.wifiConnect(WIFISSID, PASSWORD);

    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, 1);

     dht.begin();
    sensor_t sensor;
    dht.temperature().getSensor(&sensor);
    delayMS = sensor.min_delay / 1000;
}
```

Nota. Fuente: Elaboración propia (2021)

Por último, se realiza el código de ejecución, el cual definirá la manera como el microprocesado ejecutará el trabajo y repetirá en bucle.

Figura 49

Programa de ejecución del código de procesamiento

```
void loop()
{
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  Serial.print(F("Temperature: "));
    Serial.print(event.temperature);
    Serial.println(F("°C"));
    client.add(ID_TEMP, event.temperature);
    client.send();
    delay(5000);

    dht.humidity().getEvent(&event);
    Serial.print(F("Humidity: "));
    Serial.print(event.relative_humidity);
    Serial.println(F("%"));
  client.add(ID_HUM, event.relative_humidity);
    client.send();
    delay(3000);
    client.add(ID_PH, 6);
    client.send();
    delay(3000);
}
```

Nota. Fuente: Elaboración propia (2021)

Definido todo el código del programa, se procede a cargar y validad el código.

Figura 50

Carga aprobada de código al módulo Wifi

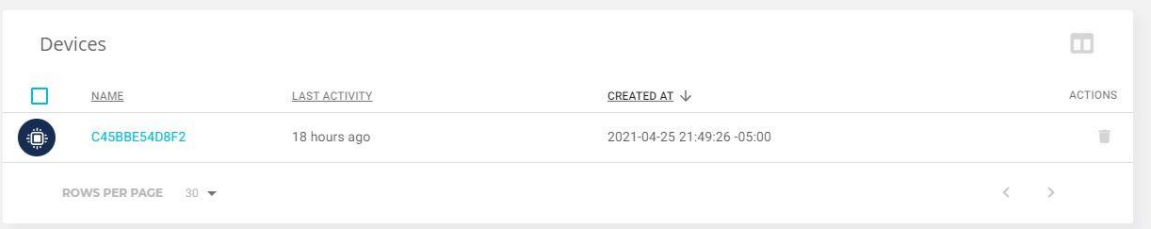
```
Executable segment sizes:
IROM   : 237624          - code in flash         (default or ICACHE_FLASH_ATTR)
IRAM   : 27616   / 32768 - code in IRAM          (ICACHE_RAM_ATTR, ISRs...)
DATA   : 1248  )         - initialized variables (global, static) in RAM/HEAP
RODATA : 744   ) / 81920 - constants             (global, static) in RAM/HEAP
BSS    : 25000 )         - zeroed variables      (global, static) in RAM/HEAP
El Sketch usa 267232 bytes (61%) del espacio de almacenamiento de programa. El máximo es 434160 bytes.
Las variables Globales usan 26992 bytes (32%) de la memoria dinámica, dejando 54928 bytes para las variables locales. El máximo es 81920 bytes.
```

Nota. Fuente: Elaboración propia (2021)

Ya cargado el código, se prepara la plataforma Ubidots para que reciba los datos que se encuentra enviando el dispositivo. Por lo que se crea primeramente el dispositivo en la plataforma tal cual como se encuentra nombrado en el código, y se registra y valida el token con el que el dispositivo y la plataforma se van a comunicar.

Figura 51

Creación de dispositivo en Ubidots.

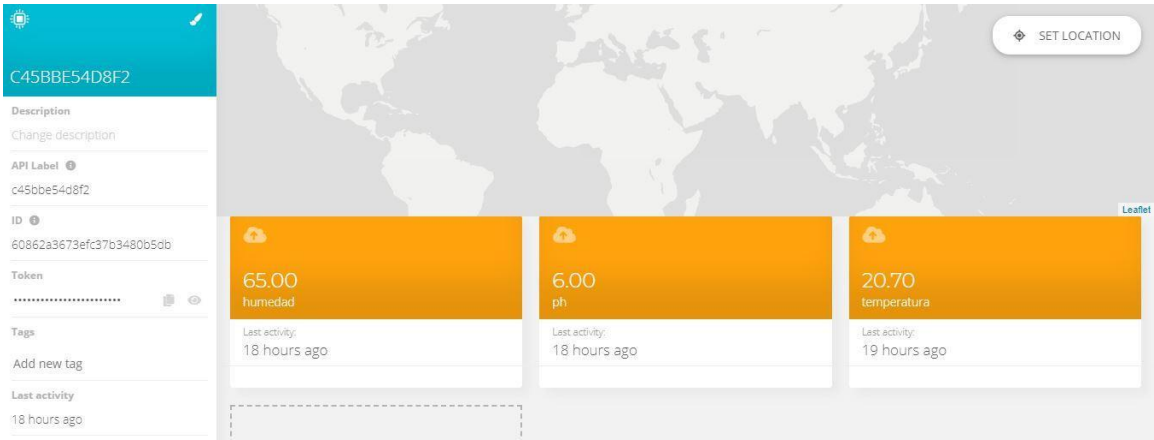


Nota. Fuente: Elaboración propia (2021)

Dentro del dispositivo creado, se procede a crear las variables de la misma manera, corroborando que los ID de variables concuerden con los de la plataforma.

Figura 52

Creación de variables en Ubidots.



Nota. Fuente: Elaboración propia (2021)

Como se observa, el sistema se encuentra enviando los datos sensados y registrados hacia la plataforma.

7.5. Pruebas

A continuación, en la tabla 24, se relacionan las pruebas ejecutadas que se a fin de validar aspectos de calidad y seguridad.

Tabla 30

Herramientas para la realización de pruebas

Prueba	Herramienta	Prueba
Calidad		
Latencia	Postman	Tiempo de respuesta en consultar estado de una planta
Mantenibilidad	SonarCloud	Reducción de code smells, bugs y vulnerabilidades
Disponibilidad	Jmeter	Número máximo de peticiones
Seguridad		
Reconocimiento	Nmap	Verificación de puertos y detección del sistema operativo del componente web del sistema
Vulnerabilidad	Owasp ZOP	Escaneo de vulnerabilidades del componente web del sistema

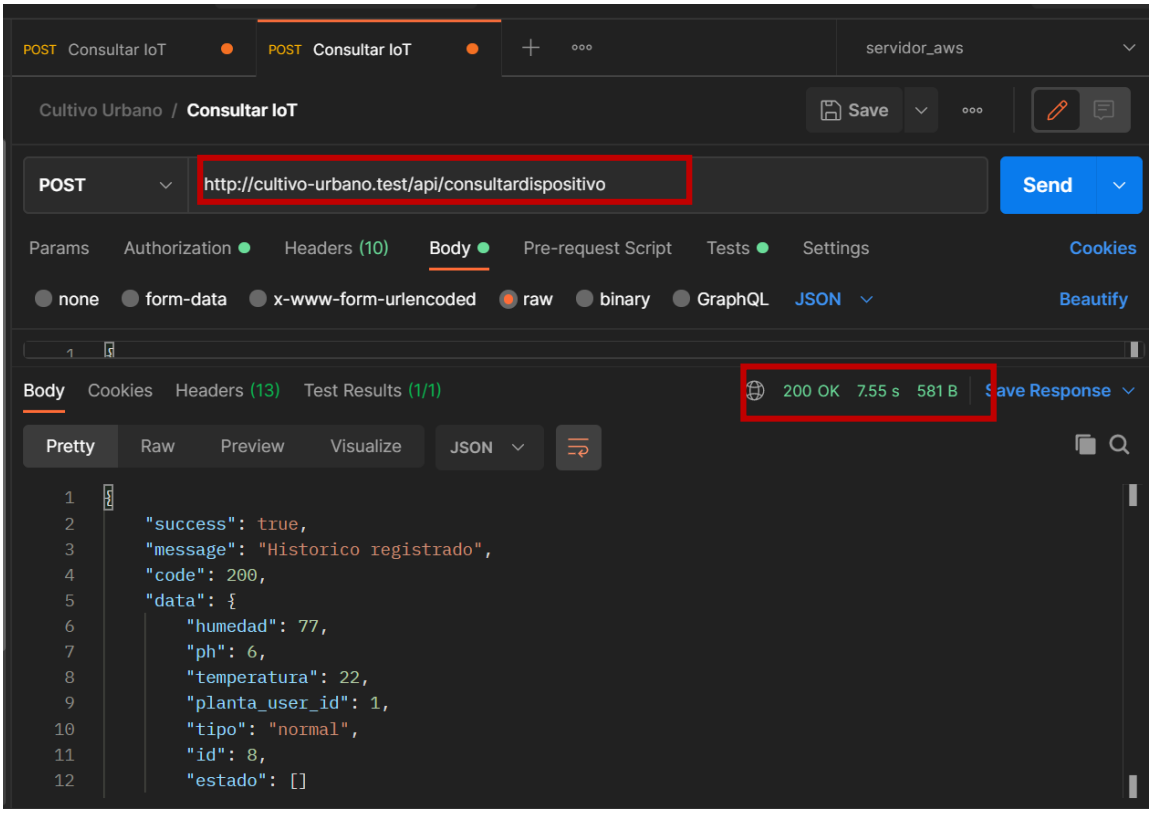
Fuente Elaboración propia (2021)

Las pruebas mencionadas se aplicarán al endpoint de consulta de datos de la planta dado que esta es la funcionalidad que más valor representa en el proyecto sitio http://cultivo-urbano.dagodigital.com/api/consultardispositivo.

En cuanto a latencia, en la figura 53 se visualiza una petición de consulta del dispositivo sensor en el entorno local de desarrollo que tardo 7.55 segundos.

Figura 53

Consulta a sensor en entorno local de desarrollo.

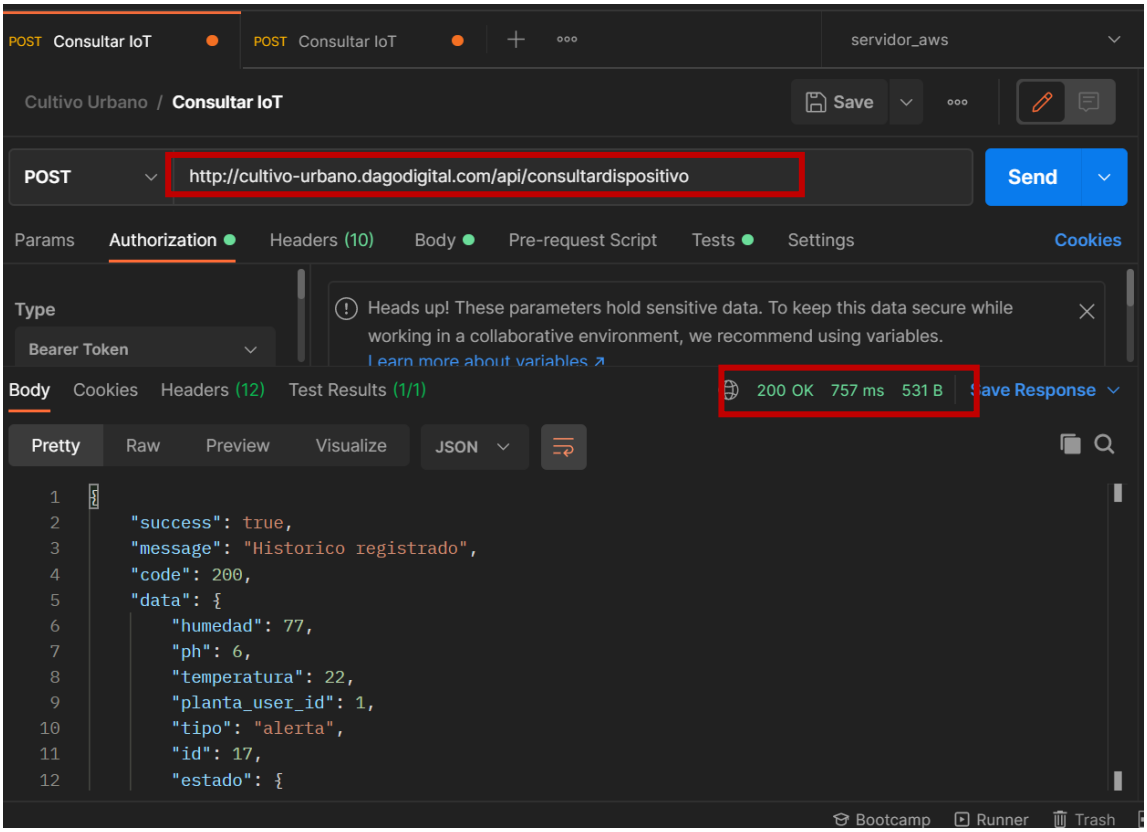


Nota. Fuente: Elaboración propia (2021)

Sin embargo, la misma petición realizada al entorno de producción desplegado en una instancia ec2 del servicio de AWS En la url http://cultivo-urbano.dagodigital.com/ tardo 757ms como se observa en la figura 54.

Figura 54

Consulta a sensor en entorno de producción en AWS

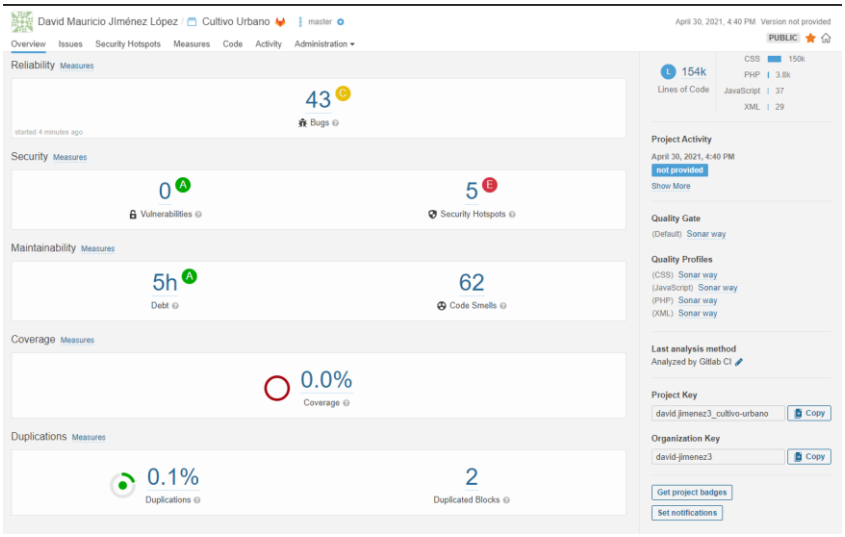


Nota. Fuente: Elaboración propia (2021)

En cuanto a la mantenibilidad, definida este estilo de código se realiza una verificación de este a través de SonarCloud para verificar aspectos de calidad del código como vulnerabilidades, índice de mantenibilidad, code smells, entre otros, en la figura 55 se visualiza un reporte de análisis inicial del análisis estático.

Figura 55

Reporte inicial análisis estático Sonar Cloud

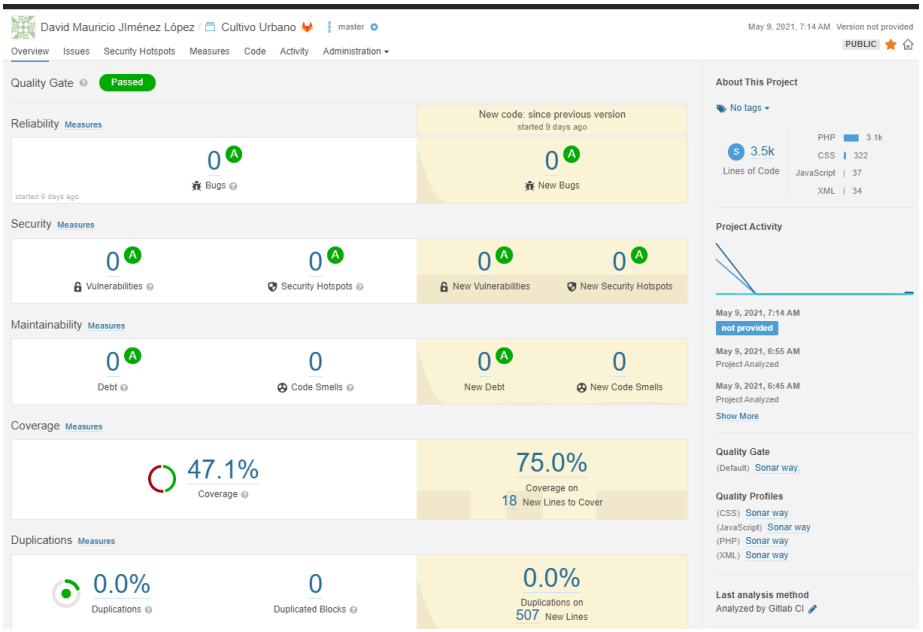


Nota. Fuente: Elaboración propia (2021)

En la figura 56 se observa el análisis estático final

Figura 56

Reporte final análisis estático Sonar Cloud

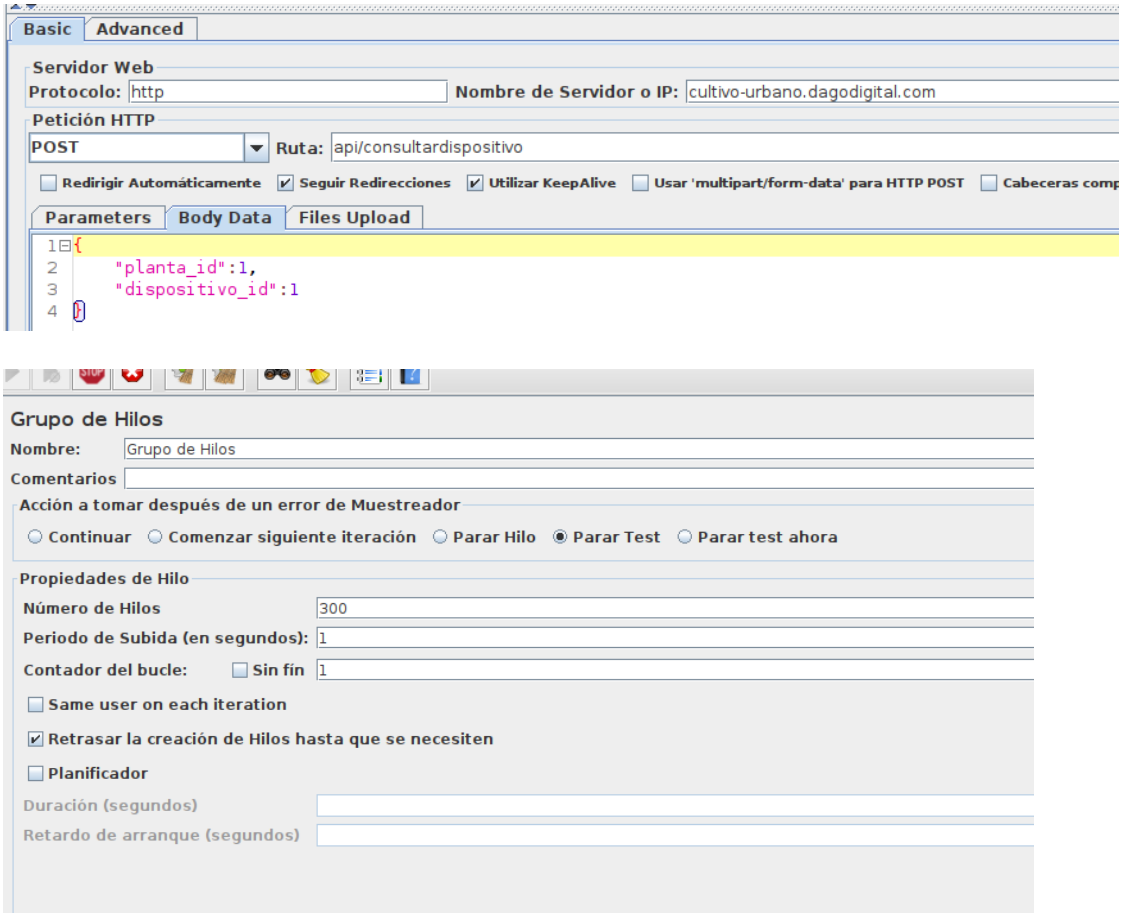


Nota. Fuente: Elaboración propia (2021)

Para evaluar la disponibilidad se realizan pruebas de estrés. A fin de determinar la capacidad de respuesta y por tanto la disponibilidad de los datos capturados por los sensores teniendo en cuenta que el backend fue desplegado en una instancia gratuita de AWS cuyas características son

- Numero de CPU's: 1
- Memoria 1 GB:
- Procesador: Intel 3.3Ghz
- Rendimiento de red : De bajo a moderado

Figura 57 *Configuración de pruebas con JMeter*



Nota. Fuente: Elaboración propia (2021)

Al ejecutar las pruebas, se dan los resultados relacionados en las imágenes 58,59 y 60

Figura 58

Detalle de peticiones realizadas con JMeter.

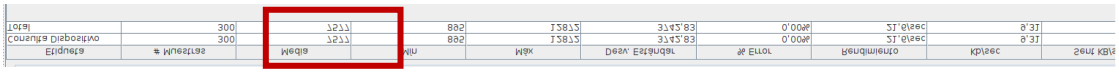
Muestra #	Tiempo de comienzo	Nombre del hilo	Etiqueta	Tiempo de Muestra (...	Estado	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	10:08:12.816	Grupo de Hilos 1-2	Consulta Dispositivo	958	✓	415	121	958	170
2	10:08:12.879	Grupo de Hilos 1-20	Consulta Dispositivo	895	✓	415	121	895	90
3	10:08:12.854	Grupo de Hilos 1-13	Consulta Dispositivo	926	✓	415	121	926	132
4	10:08:12.826	Grupo de Hilos 1-5	Consulta Dispositivo	954	✓	415	121	954	160
5	10:08:12.872	Grupo de Hilos 1-18	Consulta Dispositivo	908	✓	415	121	908	99
6	10:08:12.875	Grupo de Hilos 1-19	Consulta Dispositivo	1015	✓	440	121	1015	97
7	10:08:12.868	Grupo de Hilos 1-17	Consulta Dispositivo	1052	✓	440	121	1052	104
8	10:08:12.813	Grupo de Hilos 1-1	Consulta Dispositivo	1108	✓	440	121	1108	159
9	10:08:12.850	Grupo de Hilos 1-12	Consulta Dispositivo	1413	✓	415	121	1412	136
10	10:08:12.861	Grupo de Hilos 1-15	Consulta Dispositivo	1402	✓	415	121	1402	125
11	10:08:12.836	Grupo de Hilos 1-8	Consulta Dispositivo	1427	✓	415	121	1427	150
12	10:08:12.847	Grupo de Hilos 1-11	Consulta Dispositivo	1416	✓	537	121	1416	131
13	10:08:12.833	Grupo de Hilos 1-7	Consulta Dispositivo	1430	✓	415	121	1430	153
14	10:08:12.840	Grupo de Hilos 1-9	Consulta Dispositivo	1423	✓	537	121	1423	141
15	10:08:12.864	Grupo de Hilos 1-16	Consulta Dispositivo	1400	✓	537	121	1400	117
16	10:08:12.857	Grupo de Hilos 1-14	Consulta Dispositivo	1436	✓	440	121	1435	129
17	10:08:12.819	Grupo de Hilos 1-3	Consulta Dispositivo	1478	✓	440	121	1477	167

Nota. Fuente: Elaboración propia (2021)

En la figura 59 se encuentra que, si bien las peticiones son procesadas de manera exitosa de acuerdo con la columna estado, la latencia aumenta de forma exponencial.

Figura 59

Resumen de prueba con JMeter



Nota. Fuente: Elaboración propia (2021)

En el resumen se puede reflejar la latencia promedio en la columna Media

Figura 60

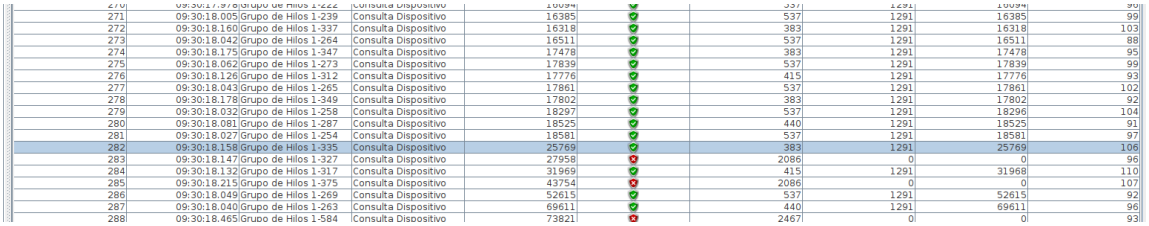
Gráfico de rendimiento de las pruebas con JMeter

Nota. La línea de rendimiento es un aproximado realizado por el autor dado que JMeter no presenta una buena resolución en las gráficas. Fuente. Elaboración propia (2021)

A fin de someter al sistema al límite, se realiza otra prueba con 1000 usuarios concurrentes, los resultados se relacionan en las figuras 61, 62 y 63

Figura 61

Detalle de peticiones en condiciones de estrés.



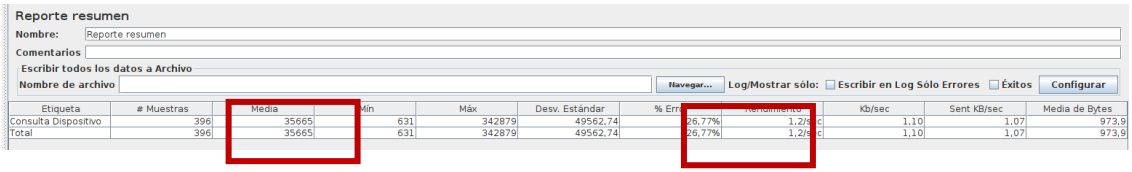
271	09:30:18.005	Grupo de Hilos 1-239	Consulta Dispositivo	16385	✓	537	1291	16385	99
272	09:30:18.160	Grupo de Hilos 1-337	Consulta Dispositivo	16318	✓	383	1291	16318	103
273	09:30:18.042	Grupo de Hilos 1-264	Consulta Dispositivo	16511	✓	537	1291	16511	88
274	09:30:18.175	Grupo de Hilos 1-347	Consulta Dispositivo	17478	✓	383	1291	17478	95
275	09:30:18.062	Grupo de Hilos 1-273	Consulta Dispositivo	17839	✓	537	1291	17839	99
276	09:30:18.126	Grupo de Hilos 1-312	Consulta Dispositivo	17776	✓	415	1291	17776	93
277	09:30:18.043	Grupo de Hilos 1-265	Consulta Dispositivo	17861	✓	537	1291	17861	102
278	09:30:18.178	Grupo de Hilos 1-349	Consulta Dispositivo	17802	✓	383	1291	17802	92
279	09:30:18.032	Grupo de Hilos 1-258	Consulta Dispositivo	18297	✓	537	1291	18296	104
280	09:30:18.081	Grupo de Hilos 1-287	Consulta Dispositivo	18525	✓	440	1291	18525	91
281	09:30:18.027	Grupo de Hilos 1-254	Consulta Dispositivo	18581	✓	537	1291	18581	97
282	09:30:18.158	Grupo de Hilos 1-335	Consulta Dispositivo	25769	✓	383	1291	25769	106
283	09:30:18.147	Grupo de Hilos 1-327	Consulta Dispositivo	27958	✗	2086	0	0	96
284	09:30:18.132	Grupo de Hilos 1-317	Consulta Dispositivo	31969	✓	415	1291	31968	110
285	09:30:18.215	Grupo de Hilos 1-375	Consulta Dispositivo	43754	✗	2086	0	0	107
286	09:30:18.049	Grupo de Hilos 1-269	Consulta Dispositivo	52615	✓	537	1291	52615	92
287	09:30:18.040	Grupo de Hilos 1-263	Consulta Dispositivo	69611	✓	440	1291	69611	96
288	09:30:18.465	Grupo de Hilos 1-584	Consulta Dispositivo	73821	✗	2467	0	0	93

Nota. Fuente: Elaboración propia (2021)

Se puede aprecia como a partir de la petición 282 el sistema empieza a tener fallos

Figura 62

Resumen de peticiones en condiciones de estrés.



Reporte resumen

Nombre: Reporte resumen

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: | Escribir en Log Sólo Errores | Éxitos | Configurar

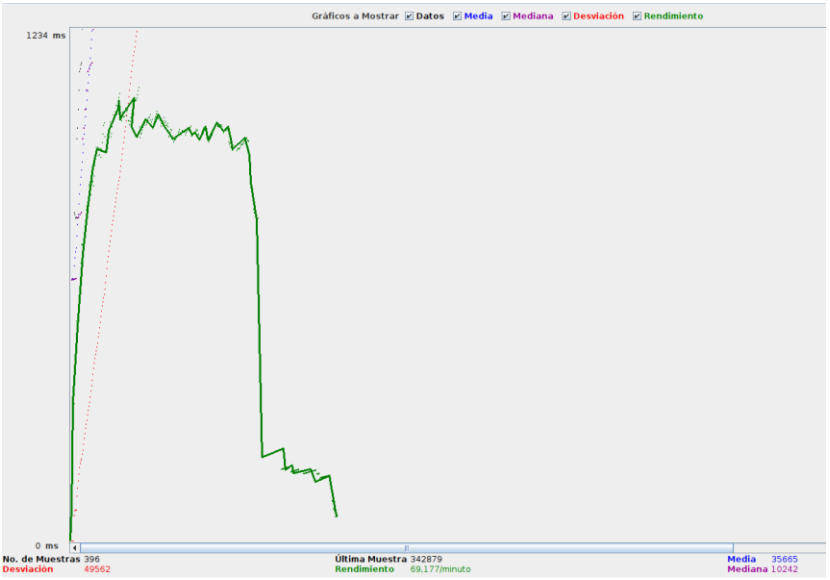
Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Err	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Consulta Dispositivo	396	35665	631	342879	49562,74	26,77%	1,2/sec	1,10	1,07	973,9
Total	396	35665	631	342879	49562,74	26,77%	1,2/sec	1,10	1,07	973,9

Nota. Fuente: Elaboración propia (2021)

A diferencia de la prueba anterior se presenta una tasa de error del 25 % y una latencia promedio de 35 segundos

Figura 63

Gráfico de rendimiento con JMeter en condiciones de estrés.

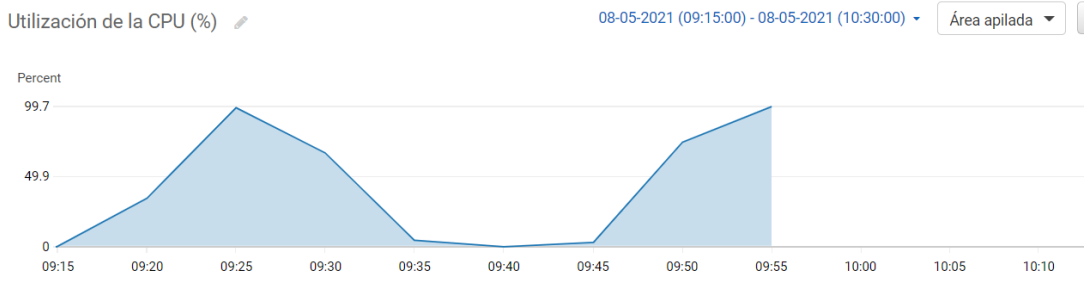


Nota. La línea de rendimiento es un aproximado realizado por el autor dado que JMeter no presenta una buena resolución en las gráficas. Fuente. Elaboración propia (2021)

Durante esta prueba, el servidor dejo de responder, al verificar el mismo se encontró una caída abrupta del mismo, dicho comportamiento se visualiza en la figura 64.

Figura 64

Consumo de la CPU durante las pruebas con JMeter

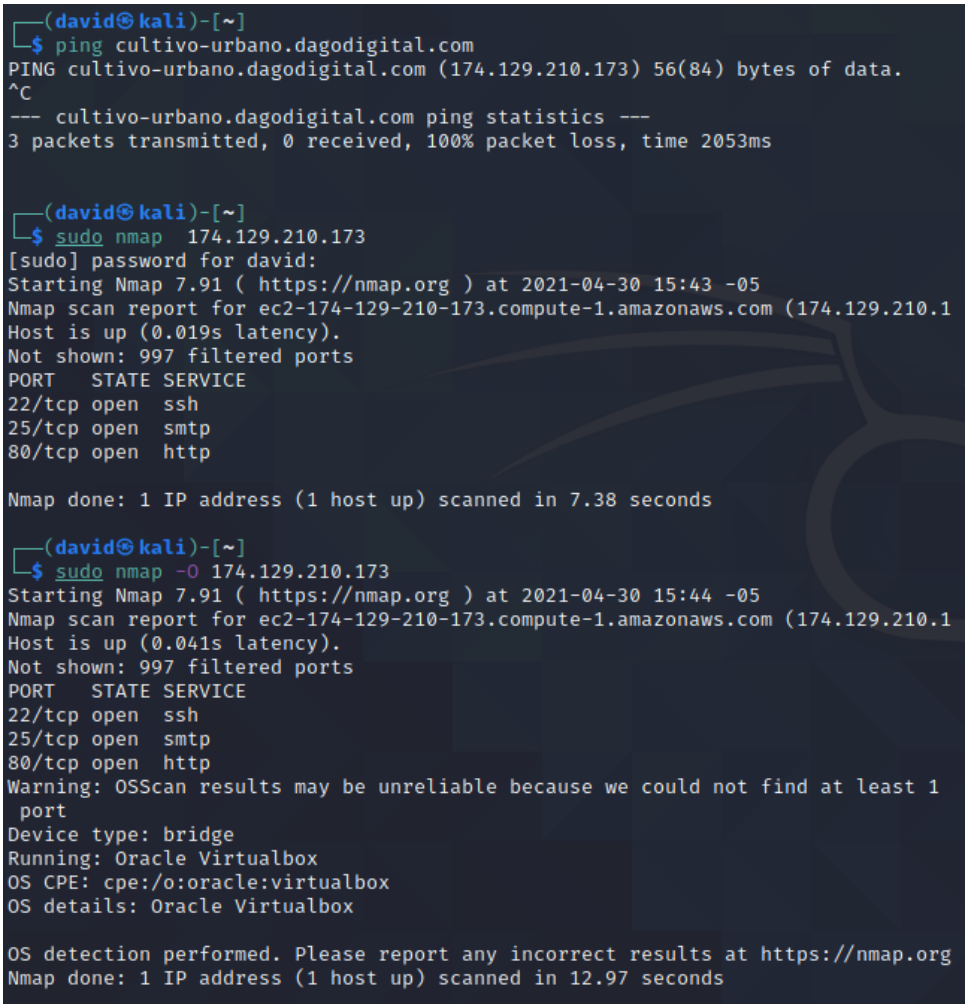


Nota. Fuente: De las 9:15 a las 9:35 es el comportamiento de la CPU iniciada la prueba con 300 usuarios concurrentes, desde las 9:40 a 9:55 es el comportamiento con la segunda prueba. Fuente. Elaboración propia (2021).

Para las pruebas de seguridad, se inicia realizando un reconocimiento a través del software Nmap como se muestra en la figura 65 con el propósito de recopilar información útil para explotar posibles vulnerabilidades

Figura 65

Reconocimiento a través de Nmap componente web.



Nota. Fuente: Elaboración propia (2021).

En la figura anterior se observa la ejecución de 3 comandos descritos a continuación:

- ping cultivo-urbano.dagodigital.com, obtenemos la dirección IP del sitio a reconocer
- sudo nmap 174.129.210.173 se obtienen los puestos abiertos y/o cerrados
- sudo nmap -O 174.129.210.173 recopila información sobre el sistema operativo,

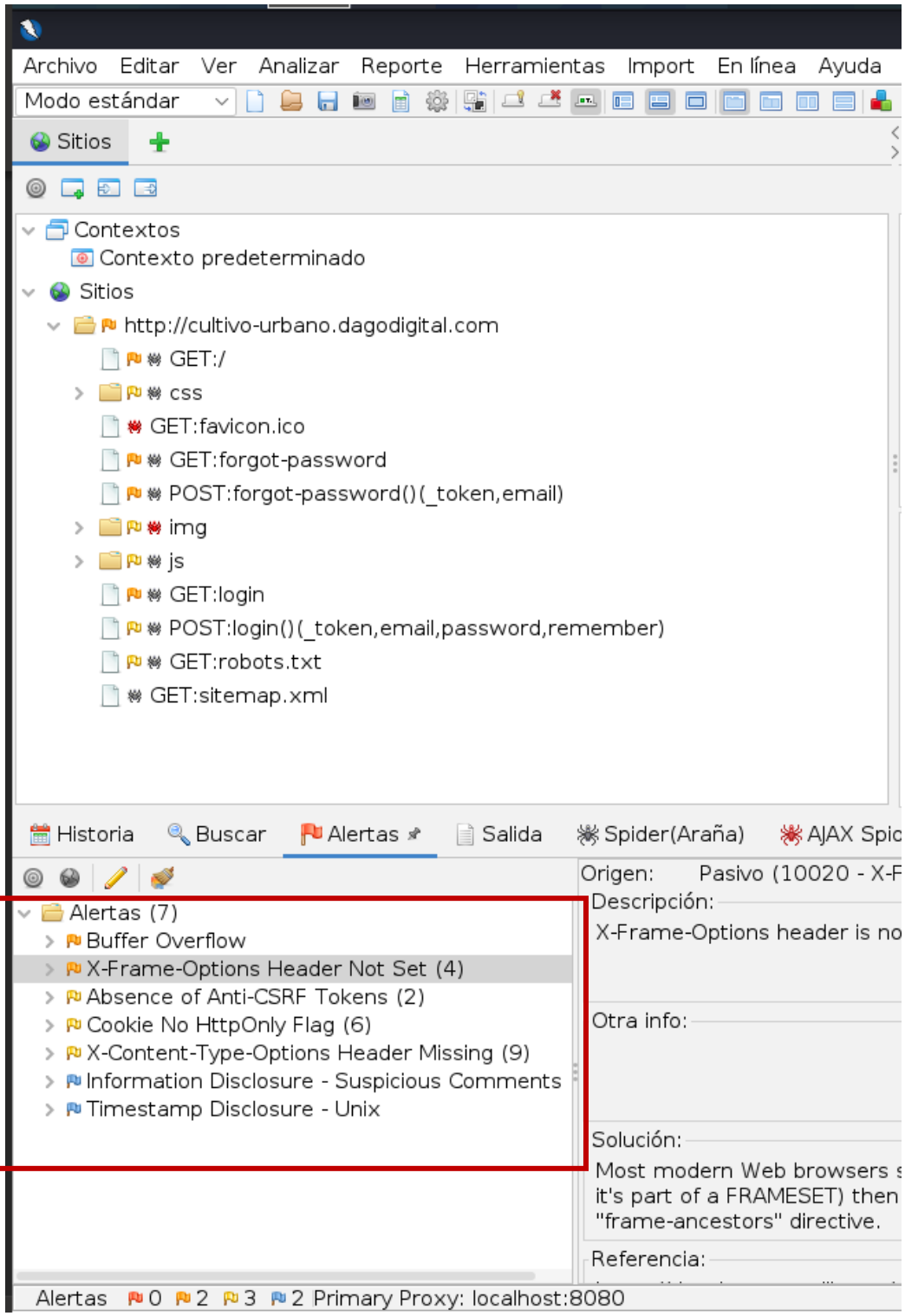
Los resultados solo mostraron 3 puertos abiertos requeridos para el desarrollo del proyecto y no se obtuvo información sobre el sistema operativo. Esto demuestra que el proveedor de infraestructura AWS ha realizado un proceso de hardering[2] de su plataforma tecnológica.

También se realiza un análisis de vulnerabilidades con la herramienta Owasp ZAP, en la figura 66 se determinan algunos hallazgos.

[2] Hardening (por su traducción al inglés endurecimiento) es el proceso reducción de vulnerabilidades en un sistema en el contexto de seguridad informática a través de la eliminación de servicios, usuarios por defecto, servicios, usuarios, etc.; innecesarios, también cerrando puertos sin uso requerido entre otras técnicas.

Figura 66

Análisis de vulnerabilidad componente Web



Nota. Fuente: Elaboración propia (2021)

De acuerdo con el cuadro de alertas, existen 2 vulnerabilidades de riesgo medio, 3 de riesgo bajo y 2 de riesgo informacional. Los riesgos identificados son controlados con las técnicas relacionadas en la tabla 31, para los riesgos bajos se sugiere su análisis en una fase posterior de desarrollo dado que el alcance del proyecto plantea un prototipo funcional del sistema de información.

Tabla 31

Mitigación de riesgos medios detectados

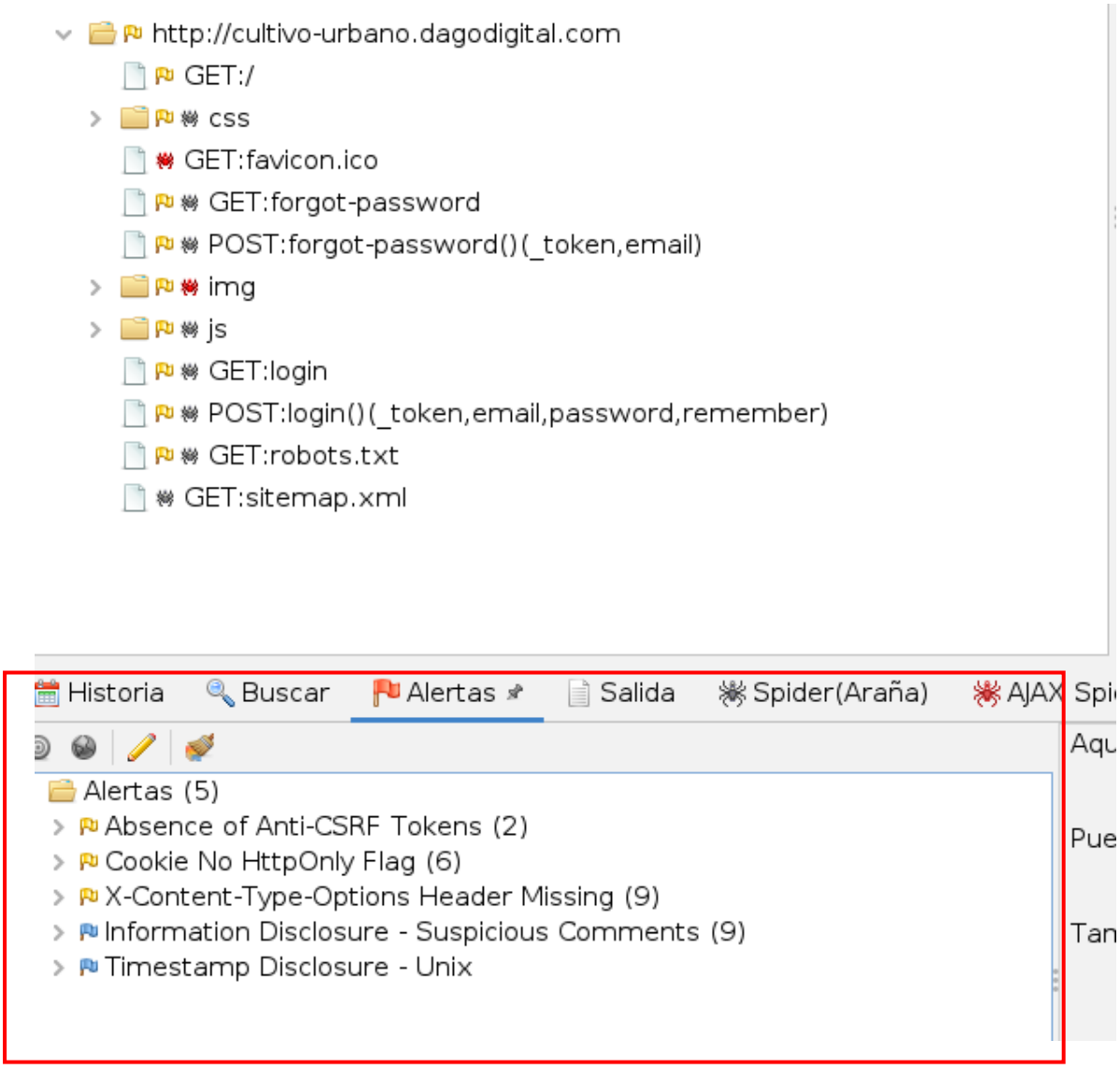
Riesgo	Medida
X-Frame-Option-Header	Se agrega el header X-Frame-Option-Header con valor 'SAMEORIGIN' a todas las respuestas del componente web.
Buffer Overflow	Se realiza validación desde el backend de los campos ingresados tanto desde el administrador web como desde la api rest implementada.

Nota. Fuente: Fuente Elaboración propia (2021)

Al realizar nuevamente el análisis se obtienen los resultados de la figura 67

Figura 67

Segundo Análisis de vulnerabilidad componente web.



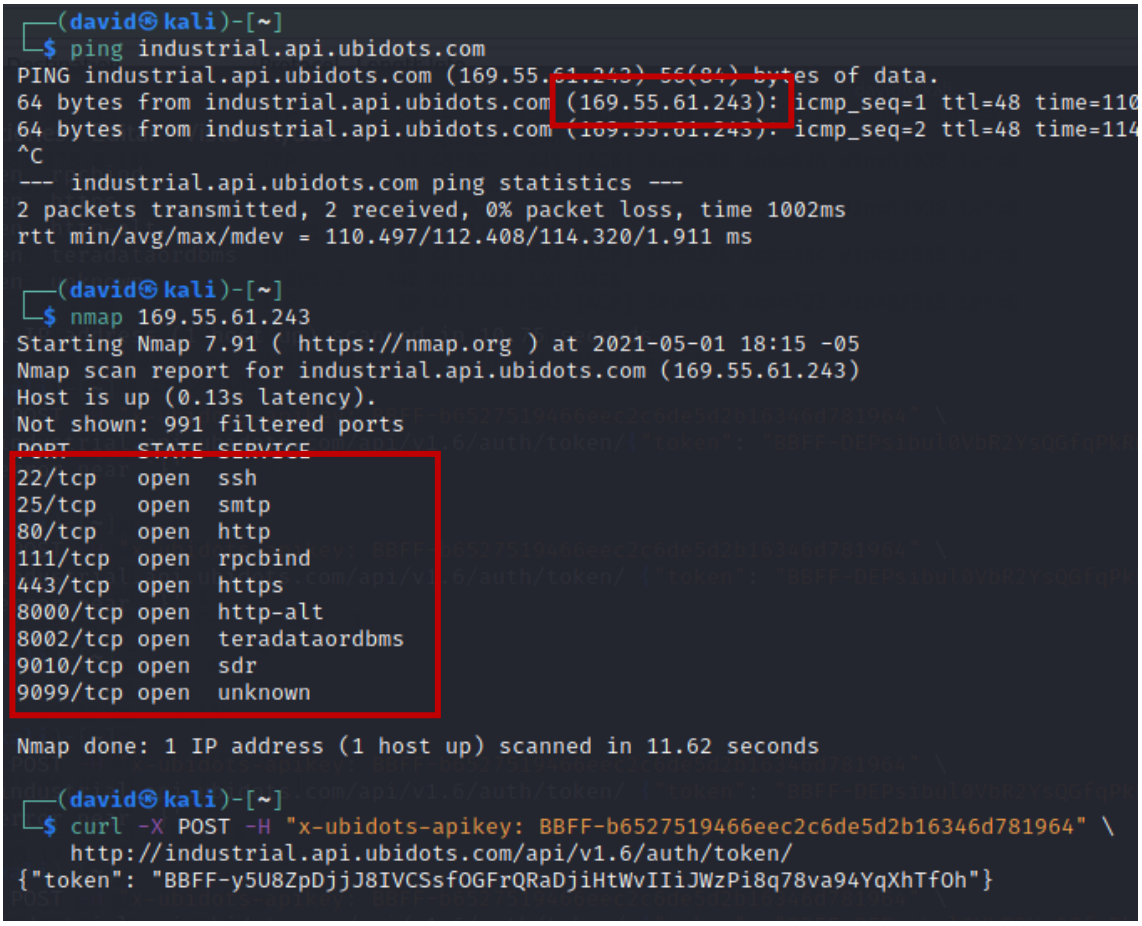
Nota. Fuente: Fuente Elaboración propia (2021)

Adicionalmente se ocultan los encabezados Server y X-Powered-By para ocultar información del servidor Web que aparecían por defecto como se muestra en la figura

También se realiza una verificación de tráfico del proveedor del cloud IoT, para ello se realiza una petición post al endpoint del proveedor para obtener nuestro token de acceso a los datos registrados por el dispositivo sensor.

Figura 68

Escaneo de puertos proveedor IoT cloud

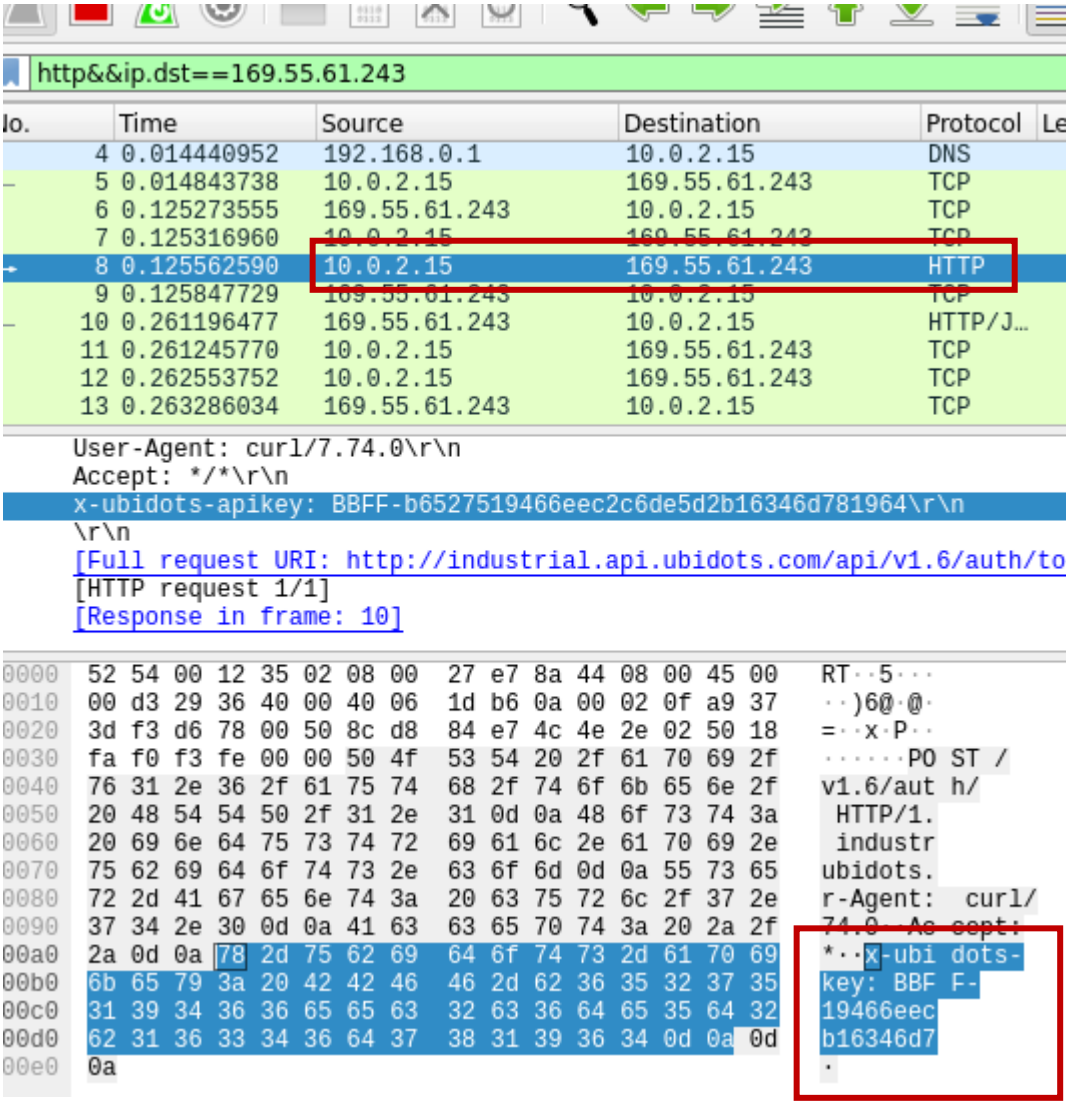


Nota. Fuente: Fuente Elaboración propia (2021)

Una vez obtenida la IP de la plataforma de IoT se verifican los puertos abiertos y se realiza una petición post por http, así se crea tráfico por el protocolo http el cual se va a monitorear con la herramienta Whireshark como se puede apreciar en la figura 69.

Figura 69

Captura de tráfico por http



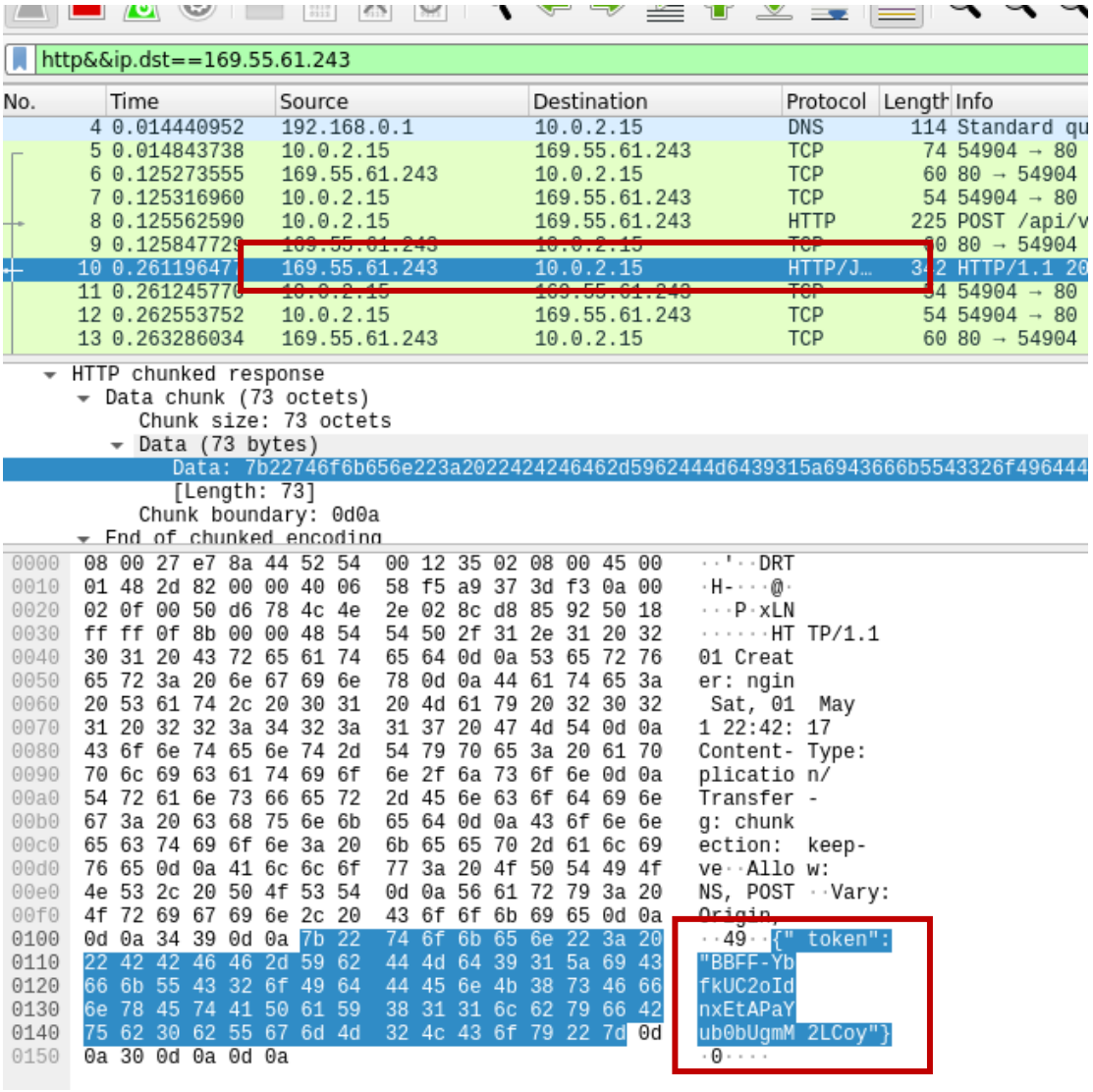
Nota. Fuente: Fuente Elaboración propia (2021)

En la parte inferior de la figura 44 se observa como el token personal para la solicitud del token de autorización es comunicado sin ningún tipo de cifrado.

En la figura 70 se observa la respuesta del proveedor de IoT donde, en el contenido del mensaje se encuentra el token de autorización recibido sin ningún tipo de cifrado.

Figura 70

Respuesta a solicitud de token de autorización



Nota. Fuente: Fuente Elaboración propia (2021)

Se encuentra que el proveedor de servicio está respondiendo a peticiones realizadas por http, razón por la cual se puede observar el token de autorización se está transmitiendo de forma clara permitiendo la captura de este por terceros.

Con el mismo procedimiento se realiza monitoreo por el puerto tcp 443 a una petición realizada por https

Figura 71

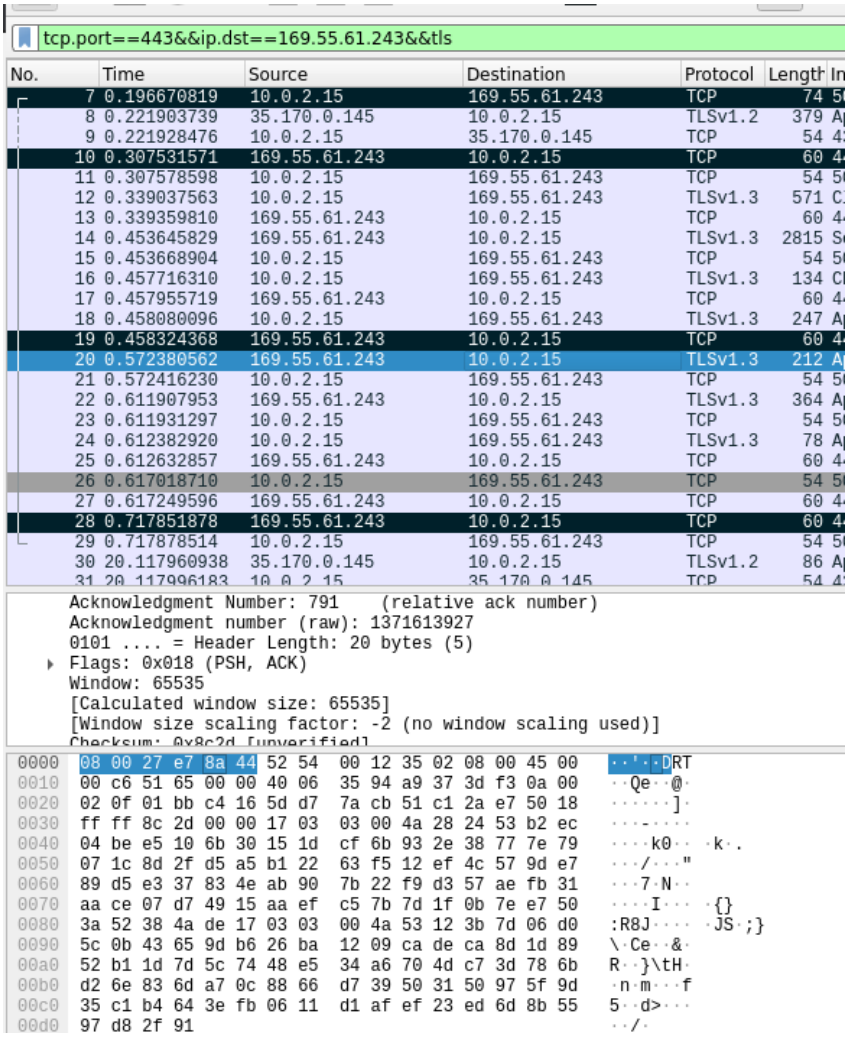
Solicitud de token de autorización por https

```
┌──(david㉿kali)-[~]
└─$ curl -X POST -H "x-ubidots-apikey: BBFF-b6527519466eec2c6de5d2b16346d781964" \
    https://industrial.api.ubidots.com/api/v1.6/auth/token/
{"token": "BBFF-z3CVMTjwuHYcodgxaRoTKDfgPdZVvlC13SB9KsXJgliXifT4OapAMYj"}
```

Nota. Fuente: Fuente Elaboración propia (2021)

Se genera el tráfico que se aprecia en la figura 73.

Figura 72 Trafico por protocolo https



Nota. Fuente: Elaboración propia (2021)

En la figura anterior se verifica el contenido de cada uno de los mensajes recibidos que figuran en negro y se encentra que el contenido aparece cifrado.

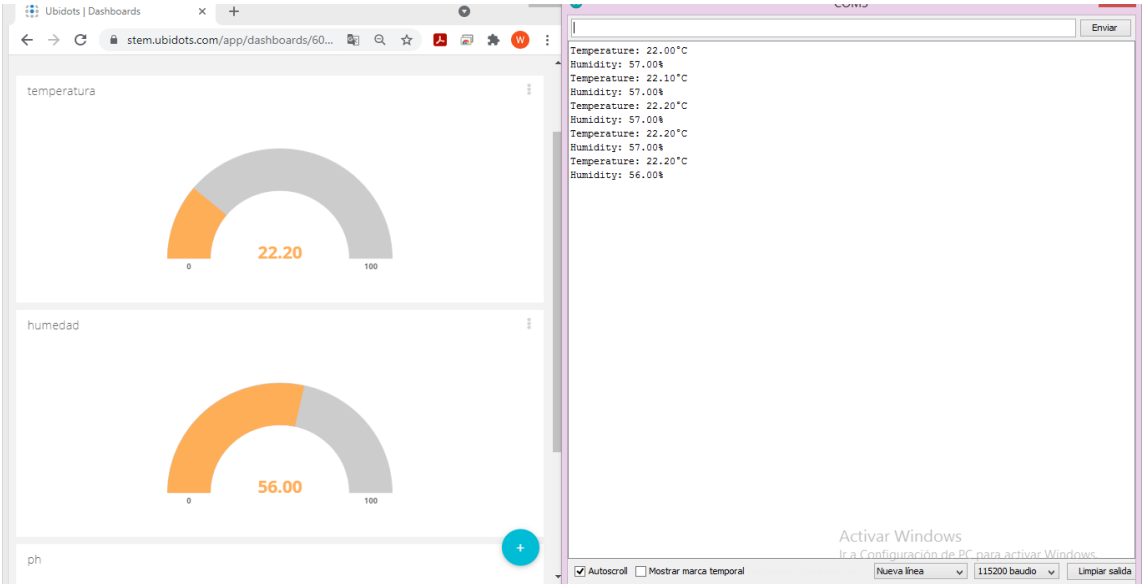
Si bien el proveedor dentro de la documentación indica el uso de https para consumir sus servicios, las evidencias anteriores demuestran que los endpoints expuestos están funcionando sobre http lo cual representa una brecha de seguridad. Es recomendable el cambio de proveedor en fases posteriores al prototipado del sistema de información.

En cuanto al Hardware, las pruebas unitarias y de integración se han de llevar a cabo junto con el proceso de construcción del código para validar su respuesta continua. Por lo que las pruebas que se realizan serán de funcionamiento continuo y su reacción a cambios.

Se procede a hacer una comparativa inicial entre los datos que registra Ubidots y los que registra el comando serial de Arduino en tiempo real.

Figura 73

Comparativa Registros Ubidots – Comando serial Arduino



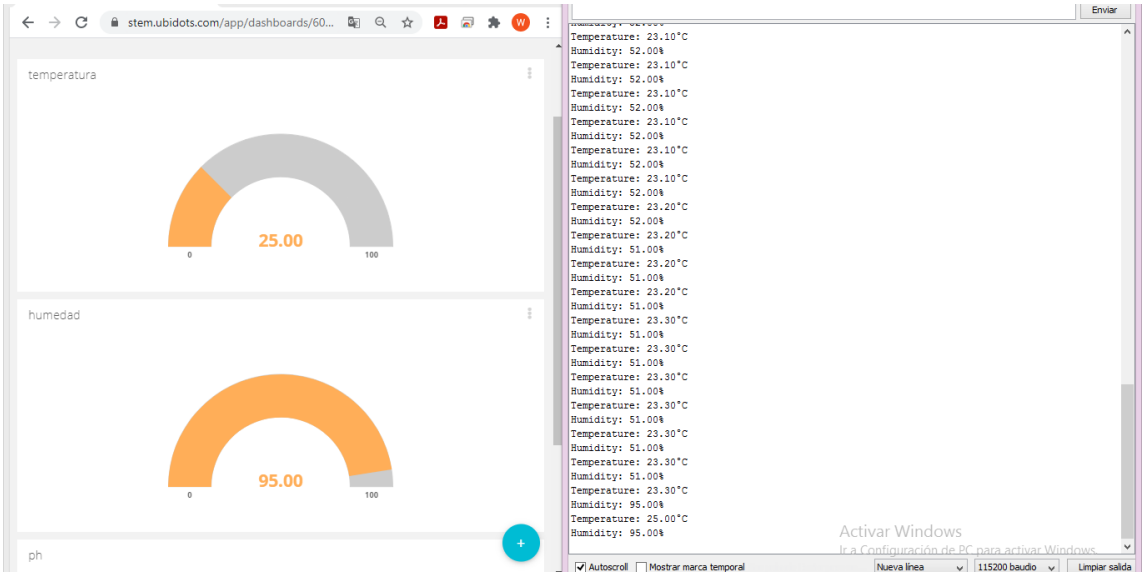
Nota. Fuente: Fuente Elaboración propia (2021)

Se registra un Delay de 600ms entre la toma del dato y el envío a la plataforma de Ubidots, lo cual es conforme a la solicitud de registro de información pactada previamente en los alcances del dispositivo.

Se procede a hacer un cambio en los parámetros del sensor para detectar su respuesta.

Figura 74

Respuesta ante cambio de ambiente del sensor



Nota. Fuente: Fuente Elaboración propia (2021)

Como se observa, el sensor registra el cambio de ambiente y se mantiene registrando los valores.

Figura 75

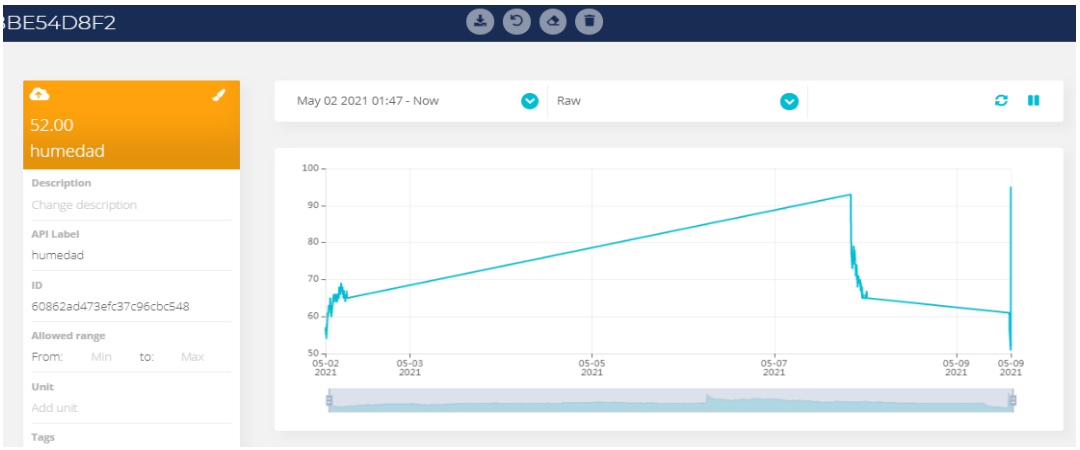
Curva de comportamiento de variable temperatura

Nota. Fuente: Fuente Elaboración propia (2021)

Se logra observar en los tramos finales de la curva, el cambio de ambiente al que se vio afectado el sensor en términos de temperatura.

Figura 76 *Curva de comportamiento de variable Humedad*

Curva de comportamiento de variable Humedad



Nota. Fuente: Fuente Elaboración propia (2021)

Se logra observar también, un comportamiento similar en la curva de humedad, validando la concordancia de los datos registrados.

Se incluye también el cálculo de tiempo de envió de información a la plataforma Ubidots.

$$T = t_u + t_T + t_h + t_{ph} + t_c$$

En donde:

$T = Periodo\ de\ registro\ de\ variables t_u$

$= Tiempo\ en\ el\ que\ el\ ubidots\ registra\ el\ dato\ conforme\ envía\ el\ sensor t_T$

$= Tiempo\ propuesto\ para\ registro\ de\ temperatura t_h$

$= Tiempo\ propuesto\ para\ registro\ de\ Humedad t_{ph}$

$= Tiempo\ propuesto\ para\ registro\ de\ ph t_c = Tiempo\ de\ procesamiento\ del\ dispositivo$

$$T = 300ms + 500ms + 500ms + 500ms + 125\mu s$$

$$T = 1.8seg$$

El tiempo de respuesta de todo el sistema es, por consiguiente, 1.8seg

7.6. Instalación y configuración

Para la instalación de la aplicación móvil se debe copiar el archivo el archivo cultivo-urbano.apk en el dispositivo móvil, asegúrese de que este permita instalar aplicaciones de desconocidas.

Para el despliegue del componente web se debe cumplir con los siguientes requerimientos en el servidor:

- Servidor Web: Apache > = 2.4 o Nginx >= 1.19.10

- PHP >= 7.2.5 con las librerías BCMath PHP Extension, Ctype PHP Extension, Fileinfo PHP extensión, JSON PHP Extension, Mbstring PHP Extension, OpenSSL PHP Extension, PDO PHP Extension, Tokenizer PHP Extension, XML PHP Extension.
- Base de datos MySQL >= 5.8 o Postgres >= 9
- Git 0> 2.01

Copiar el contenido del proyecto en las carpetas publicas configuradas en el servidor web, si el proyecto se encuentra en un repositorio de git se deben ejecutar los siguientes comandos través del método clone de git

Git clone [repositorio]

Cd [ruta a la carpeta raíz del proyecto nombre carpeta repositorio]

Una vez en la ubicación raíz del proyecto se deben conceder permisos a la carpeta storage de la siguiente forma:

sudo chown -R $USER:www-data storage

sudo chown -R $USER:www-data bootstrap/cache

sudo chmod -R 775 storage

sudo chmod -R 775 bootstrap/cache

Luego se instalan las librerías utilizadas y se procede a realizar las configuraciones con los siguientes comandos

composer install

php artisan key:generate

php artisan migrate –seed

php artisan passoport:install

php artisan config:cache

cp .env-example .env

En el archivo .env copiado se deben configurar los siguientes de la base de datos:

DB_CONNECTION=mysql // o postgress

DB_HOST=localhost

DB_PORT=3306

DB_DATABASE=db-cultivo //nombre de la base de datos previamente creada

DB_USERNAME=   //USUARIO CON PERMSIO EN LA ABSE DE DATOS

DB_PASSWORD= //CONTRASEÑA D EUUSARIO

Luego, en el mismo archivo configurar los parámetros de conexión de su servicio de correo electrónico, a continuación, se relaciona un ejemplo con mailtrap.io

MAIL_MAILER=smtp

MAIL_HOST=smtp.mailtrap.io

MAIL_PORT=2525

MAIL_USERNAME=35adffca4f5e34

MAIL_PASSWORD=ffeb584339b0a2

MAIL_ENCRYPTION=tls

MAIL_FROM_ADDRESS=administrador@cultivo-urbano.dagodigital.com

Por último , en el mismo archivo asignar los siguientes valores:

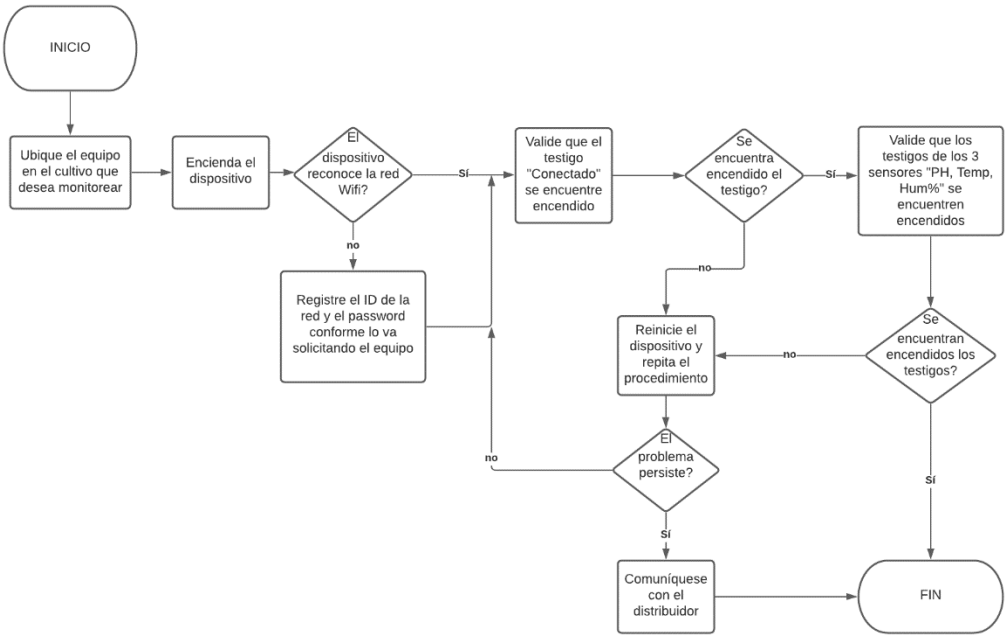
APP_NAME=Cultivo-Urbano

APP_ENV=production

APP_DEBUG=false

La instalación del sensor para su funcionamiento es bastante sencilla y se lleva a cabo bajo el diagrama de flujo de la figura 78.

Figura 77 *Instalación del sistema sensor*



Nota. Fuente: Fuente Elaboración propia (2021)

Conclusiones

Al realizar la construcción del sistema de información a partir de las funciones que dan valor agregado, en este caso la medición de variables de la planta permite una identificación oportuna de los componentes del sistema para una correcta asignación de tareas a partir de las habilidades y conocimientos de los integrantes del grupo de trabajo. A fin de crear el prototipo que da una respuesta a la problemática planteada en una fase inicial, se indago sobre las tendencias, tecnologías y herramientas para un sistema de información basado en IoT que permitieron la construcción de este a partir de la aplicación de los conocimientos adquiridos durante la especialización que fueran pertinentes al alcance del proyecto.

Un componente fundamental sobre el cual se centra el componente de innovación es el sistema de captura de datos de la planta, llevando a cabo la implementación y, basados en los requerimientos de la planta y el nivel de monitoreo que requiere, se llevaron a cabo la inclusión de los módulos propuestos previamente, ya que su rango de medición y la capacidad de trabajo continuo que poseen es bastante fiable, soportándonos de los requerimientos del sistema en cuanto a disponibilidad y la capacidad de niveles de flujo de trabajo altos. Además de esto, se abordaron varias soluciones para la etapa de registro de información en la nube, explorando diferentes periferias, pero tomando finalmente la decisión por la plataforma de Ubidots, cuya interfaz es fácil de usar y permite extracción en vivo de los datos, lo cual satisface los requerimientos del sistema en cuanto a la latencia y la tolerancia a fallos. A pesar de ser un dispositivo electrónico, se aplicaron conceptos vistos en el desarrollo de la especialización para llevar a cabo un nivel de funcionamiento y acople adecuado con la etapa de software.

Dado que el sistema de información construido se planteó como un prototipo y por tanto está sujeto a mejoras en todos sus componentes, si bien el backend y el sensor pueden tener

mejoras transparentes para el usuario encaminadas a los requerimientos no funcionales, los funcionales como la adición de nuevas variables capturadas por los sensores como movimiento o luminosidad, deben tener el menor impacto en la usabilidad de la aplicación móvil que es la entrada al sistema de información por parte de los usuarios. En este particular consideramos que es fundamental concebir un diseño, si bien básico, escalable en términos de usabilidad buscando la apropiación de las nuevas funciones con el menor impacto realizando una extensión de los menús, botones necesarios bajo el mismo esquema de navegabilidad en el uso de la aplicación.

La orquestación de todos los componentes del sistema de información no hubiese sido posible sin la definición de una arquitectura. Existen diferentes patrones y diseños y estilos arquitecturales, estos no deben ser tomados como una receta, sino como una guía donde pueden surgir arquitecturas propias que respondan al sistema a construir. Para IoT la tendencia es el uso de microservicios, pero en este proyecto se adoptó un estilo arquitectural basado en capas dadas las restricciones de tiempo, recursos con los que se contaba y la implementación de un proveedor de IoT para la persistencia de datos, razón por la cual creemos que la arquitectura no solo debe responder al producto a construir a nivel de desarrollo de software si no también al proyecto.

Aunque la arquitectura no brinda las bases para hacer un producto estable y perdurable en el tiempo, por si misma no puede asegurar la calidad de dicho producto. Para hecho la aplicación de escenarios de calidad permiten diseñar las pruebas para validar la calidad esperada del producto, es cierto que son varios los atributos de calidad que se pueden identificar en el sistema de información construido como la pertinencia funcional, coexistencia, madures, usabilidad, entre otros entre otros, consideramos que son fundamentales para el desarrollo de prototipos funcionales la mantenibilidad, dado que una vez se identifique la viabilidad de un prototipo se deben añadir mejoras, la latencia en la medida que puede identificar problemas de rendimiento

que, si no se subsanan en el prototipado, pueden replicarse a otros componentes y la disponibilidad. Este último consideramos que es prioritario de acuerdo con el valor de negocio del sistema a desarrollar. Para el caso del presente proyecto la disponibilidad de datos al momento de ser consultados era vital para considerar la viabilidad del proyecto, para otros proyectos puede que se dé más prioridad a otros atributos de calidad.

Adicionalmente un atributo clave es la seguridad, al igual que la calidad, la seguridad se debe ir afianzando en etapas tempranas del ciclo de vida de desarrollo de software. Para ello el uso de la metodología Microsoft Security Development Lifecycle (SDL) nos permitió realizar a nivel de seguridad un análisis en paralelo al desarrollo de software reduciendo posibles brechas que se puedan dar durante la creación ya sea por vulnerabilidades del código construido como de componentes de terceros.

Sistemas de información basados en IoT deben contar con un estudio de factibilidad financiera debido a que , para un óptimo funcionamiento, se debe invertir en un diseño e implementación de infraestructura de TI de alto rendimiento dado el flujo de datos a manejar, adicionalmente se requiere realizar estudios de diseño de componentes a la medida para llegar a la construcción de hardware a un precio accesible, las pruebas de estrés demostraron que un servidor web de pocos recursos no soporta una gran cantidad de usuarios concurrentes. Dichos aspectos no fueron analizados en el presente proyecto dado el alcance de este.

Referencias

FarmBot. (s.f.). *FarmBot*. Obtenido de https://farm.bot/pages/express

NewFarm. (2019). *https://newfarm.land/*. Obtenido de New Farm.

Emanuel Encizo Camacho, (2006). TekRarium, el arácnido tecnológico que provee agua y alimento en condiciones extremas. *Expeditio (16)*, p.48

P., E. (2018, December 29). Una empresa malagueña crea un macetero inteligente que se controla con una app. Retrieved August 25, 2020, from https://www.laopiniondemalaga.es/malaga/2018/12/29/empresa-malaguena-crea-macetero-inteligente/1057490.html**.**

Hayward, A. (2016, July 05). The FarmBot Genesis Brings Precision Agriculture to Your Own Backyard. Recuperado agosto 25, 2020, de https://www.smithsonianmag.com/innovation/farmbot-genesis-brings-precision-agriculture-your-own-backyard-180959603/

NewFarm Inc. (2019). FAQs. Recuperado agosto 25, 2020, de https://newfarm.land/faqs/

Acosta, A, & Leon, D. (2015). Prototipo De Control Para Un Cultivo De Tomate Cherry En Un Invernadero (tesis de pregrado). Universidad Católica de Colombia, Bogotá, Colombia.

Perafan, C. (2018). Agrodroyd. Sistema de Monitoreo para cuidado y riego de productos agrícolas en cultivos urbanos (tesis de pregrado). Universidad Católica de Colombia, Bogotá, Colombia.

OMS. Alma-Ata 1978 Atención Primaria en Salud. Ginebra, Suiza: OMS; 1978. 2

Declaración de Beijing. En: Congreso de la OMS sobre Medicina Tradicional. Beijing, China; 2008.

Giraldo, S., Bernal, M., Robayo, A., Pardo, A, Gamba L, (2015). Descripción del uso tradicional de plantas medicinales en mercados populares de Bogotá, D.C. Nova, Volumen 13, 73 – 80.

Miranda, M., Velázquez, D., Bermúdez, A. (2005). La investigación etnobotánica sobre plantas medicinales, Interciencia: Revista de ciencia y tecnología de América, Volumen 30, 453-459

Dinero. (2018, noviembre 01). ¿Qué pasa con la calidad del sistema de salud en Colombia? recuperado septiembre 11, 2020, de https://www.dinero.com/pais/articulo/cual-es-el-problema-del-sistema-de-salud-colombiano/263731

Ministerio de Salud y Protección Social. (2018). lineamientos técnicos para la articulación de las medicinas y las terapias alternativas y complementarias, en el marco del sistema general de seguridad social en salud, Bogotá, Colombia

Gerber, A. (2017, octubre 4). Simplifique el desarrollo de sus soluciones de IoT con arquitecturas de IoT. Recuperado septiembre 18, 2020, de https://developer.ibm.com/es/technologies/iot/articles/iot-lp201-iot-architectures/

Foundation, E. (2017, diciembre). White Paper: The Three Software Stacks Required for IoT Architectures: IoT development made simple. Recuperado septiembre 18, 2020, de https://iot.eclipse.org/community/resources/white-papers/iot-architectures/

Texas Instruments, product folder. (2017, December). LM35 Precision Centigrate Temperature Sensors. Recuperado septiembre 18, 2020, de https://www.ti.com/lit/ds/symlink/lm35.pdf

Mouser Electronics, Data Sheet. (2016, August). DHT11 Humidity & Temperature Sensor. Recuperado septiembre 18, 2020, de https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

ElecFreaks. (2016, August). Octopus Soil Moisture Sensor Brick. Recuperado septiembre 18, 2020, de https://www.elecfreaks.com/octopus-soil-moisture-sensor-brick.html

STM Electronics. (Rev. 2020, August). Arm® Cortex®-M4 32b MCU+FPU. Recuperado septiembre 18, 2020, de https://www.st.com/resource/en/datasheet/dm00037051.pdf

Trello. Janet Mesh (2020, March). Metodología Kanban, revoluciona la manera de trabajar más ágil. Recuperado septiembre 18, 2020, https://blog.trello.com/es/metodologia-kanban

Team, T. (2018, February 5). What is Arduino? Recuperado septiembre 24, 2020, de https://www.arduino.cc/en/Guide/Introduction

Microsoft. (2018) SDL Process Guidance Version 5.2