UNIVERSIDAD CATÓLICA SANTO TORIBIO DE MOGROVEJO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

SMART GREENHOUSE APLICANDO LAS IoT PARA LA CONSERVACIÓN DEL CULTIVO DE TOMATES NATIVOS EN LA REGIÓN LAMBAYEQUE

TESIS PARA OPTAR EL TÍTULO DE
INGENIERO DE SISTEMAS Y COMPUTACIÓN

AUTOR

PAOLA PATRICIA CASTRO FERNANDEZ

ASESOR

MARIANA CHAVARRY CHANKAY

https://orcid.org/0000-0001-5136-7177

Chiclayo, 2023

SMART GREENHOUSE APLICANDO LAS IoT PARA LA CONSERVACIÓN DEL CULTIVO DE TOMATES NATIVOS EN LA REGIÓN LAMBAYEQUE

PRESENTADA POR:

Paola Patricia Castro Fernandez

A la Facultad de Ingeniería de la
Universidad Católica Santo Toribio de Mogrovejo
para optar el título de

INGENIERO DE SISTEMAS Y COMPUTACIÓN

APROBADA POR:

Maria Ysable Aranguiri Garcia
PRESIDENTE

Karla Cecilia Reyes Burgos
SECRETARIA

Mariana Chavarry Chancay
VOCAL

DEDICATORIA

El presente proyecto es dedicado a mi familia y amigos quienes ha sido siempre mi apoyo incondicional.

AGRADECIMIENTOS

El agradecimiento de este proyecto va dirigido primeramente a Dios, a mi familia y amigos quienes me dieron el apoyo emocional necesario para poder seguir adelante.

INFORME DE TESIS

INFORME DE ORIGINALIDAD

18%	18%	2%	%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	naylampmechatronics.com Fuente de Internet	3%
2	www.naylampmechatronics.com Fuente de Internet	3%
3	hdl.handle.net Fuente de Internet	2%
4	tesis.usat.edu.pe Fuente de Internet	1%
5	www.coursehero.com Fuente de Internet	1%
6	idoc.pub Fuente de Internet	1%
7	dspace.ups.edu.ec Fuente de Internet	1%
8	redi.unjbg.edu.pe Fuente de Internet	1%
9	bioseguridad.minam.gob.pe Fuente de Internet	<1%

Índice

Lista de evidencias

Lista de tablas

Lista de gráficos

Lista de figuras

Resumen

El tomate nativo peruano forma parte de la agrobiodiversidad del Perú, actualmente su conservación y producción la llevan a cabo los agricultores, quienes lo acogen como parte de sus otros cultivos luego de que las semillas de estas especies de tomate nativas caen en sus chacras o huertos. La preocupación que se tiene es que debido a la falta de cultivo y a las condiciones climáticas del Perú estas no se conserven adecuadamente. Es por lo que, a raíz de esta problemática, se desarrolló una investigación de tipo experimental, mediante la implementación de un Sistema IoT, con el fin de conservar mejor estas especies en un ambiente que ofrezca mayor control y monitoreo en tiempo real. Este consistió en el desarrollo de un circuito el cual fue colocado dentro de la Greenhouse, el cual tenía conectado sensores y actuadores, este emitía los datos captados por los sensores y el estado de los actuadores hacia un aplicativo móvil. Este proyecto se realizó con un cultivo de 5 ejemplares de Solanum lycopersicum variedad Cerasiforme, con el cual se obtuvo un rendimiento mayor de un 4.17% por parte del grupo experimental.

Palabras Clave: Smart Greenhouse, IoT, Tomate native peruano.

Abstract

The Peruvian native tomato is part of the agrobiodiversity of Peru, currently its conservation and production is carried out by farmers, who welcome it as part of their other crops after the seeds of these native tomato species fall on their farms or orchards. The concern is that due to the lack of cultivation and the climatic conditions of Peru, these are not properly preserved. That is why, as a result of this problem, an experimental type of research was demonstrated, through the implementation of an IoT System, in order to conserve better species in an environment that offers greater control and monitoring in real time. This will consist of the development of a circuit which was placed inside the Greenhouse, which had sensors and actuators connected, this would emit the data captured by the sensors and the status of the actuators to a mobile application. This project was carried out with a culture of 5 specimens of Solanum lycopersicum variety Cerasiforme, with which a yield greater than 4.17% was obtained by the experimental group.

Keywords: Smart Greenhouse, IoT, Peruvian native tomato.

I. INTRODUCCIÓN

El cultivo de tomate (Solanum lycopersicum L.) es mundialmente realizado y conocido siendo una de las hortalizas más demandadas a nivel mundial; en el Perú es cosechada con mayor frecuencia en la costa, especialmente en ciudades como Lima e Ica, sin embargo, el tomate nativo peruano es cultivado en menor cantidad debido a la pobreza extrema e inseguridad alimentaria que existe en los agricultores tradicionales, los cuales representan a la población más vulnerable ante eventos extremos ocasionados por el cambio climático. El tomate nativo peruano forma parte de la agro diversidad del Perú, cuya conservación actualmente es llevada a cabo por los productores quienes no lo mantienen necesariamente cultivándolo, tampoco lo tienen en cantidad; la forma como se tienen en sus chacras o huertos es porque las semillas caen a la chacra, y luego la planta desarrollada pasa al cuidado del agricultor, formando parte de sus otros cultivos. Tomando en cuenta como es llevado a cabo el cultivo de estas especies propias del Perú convierte un reto el armonizar la modernidad y el desarrollo con la conservación de los tomates nativos [1].

Se ha constatado en el Perú un total de 13 especies de tomate, 12 silvestres y una cultiva, tenido una amplia presencia el tomate nativo (S. lycopersicum variedad cerasiforme) mejor conocido como Tomate Cereza o Cherry. Se espera fortalecer el desarrollo de tomates nativos con acciones de conservación in situ y ex situ, tal y como lo expresa el MINAGRI [2]. El Perú es considerado como uno de los países que más se ven afectados con el cambio climático según el MINAM, siendo propenso a diversos tipos de efectos causado por el mismo tales como son las heladas, friaje, plagas e inundaciones, siendo más propensas en la costa las dos últimas mencionadas debido al clima habitual de la zona [3].

La necesidad de implementar nuevas tecnologías se ve en los riesgos que causa el cambio climático en los cultivos. Los cambios climáticos, no son totalmente predecibles, y los datos del ambiente en el cual se desarrolla un cultivo son importantes para saber, si este podrá llegar a ser exitoso. En los últimos años una de las tendencias más usadas por muchos agricultores para disminuir los posibles percances causados por el clima y las plagas son los cultivos bajo techo, sea mediante casa maya, invernadero o macro túneles [4].

Es importante acunar nuevas ideas para mejorar e innovar en los cultivos de tomates nativos, debido a que la mayoría de estas especies nativas se encuentran en peligro de perderse debido a la falta de cultivo, especialmente tomando en cuenta los cambios climáticos que se han estado experimentando en sus zonas de cultivo. Por lo mismo llegamos a la pregunta ¿Cómo se podría controlar y monitorear el ambiente en cultivos de tomates nativos en la región de Lambayeque para contribuir a su conservación?

Este proyecto tiene como objetivo desarrollar una solución basada en IoT, el cual traducido al en español se conoce como internet de las cosas, las cuales hacen referencia a una red de objetos físicos que permiten el intercambio de datos entre dispositivos mediante internet, estos objetos tienen integrados sensores, actuadores, software, entre otras tecnologías [5], las cuales harían posible la realización de un microclima controlado con acceso a una monitorización constantes que permita un mejor desarrollo en el cultivo de tomates nativos peruanos.

II. REVISIÓN DE LA LITERATURA

En este apartado se muestran los antecedentes nacionales e internacionales encontrados, así como el desarrollo de las bases teóricas de la investigación, lo cual es necesario para entender la investigación.

2.1. Antecedentes

Se han considerado para esta investigación los siguientes antecedentes nacionales e internacionales, no se encontraron antecedentes locales por los cuales estos no fueron considerados como un ítem:

2.1.1. Antecedentes Nacionales

A continuación, se presentarán los antecedentes nacionales encontrados:

- Una investigación perteneciente a la "Pontificia Universidad Católica del Perú" en la ciudad de Lima, se desarrolló un sistema para el monitoreo y control de la humedad en un invernadero, para ello se desarrolló e implemento un programa basado en la lógica ON/OFF, el cual utilizo un algoritmo de control programando en el lenguaje C, además el uso del sensor de humedad, estableciendo una comunicación serial entre el sistema de control y la computadora mediante la interfaz de usuario, dado como resultado que los actuadores se activaban de acuerdo a lo establecido en la lógica de control. Lográndose cierta autonomía en el invernadero debido a que el sistema mantenía el grado de humedad adecuado sin depender de un operario [6].

- En la "Universidad Nacional de Huancavelica" perteneciente a la ciudad de Huancavelica, se realizó una investigación en la cual consistía en el control de la humedad relativa que se encontraba en el aire en el interior de un invernadero dedicado al cultivo de tomates, para ello utilizaron un control lógico programable que activaba los humidificadores y deshumidificadores instalados en el invernadero, además también se hizo uso de sensores HSM-20G para la medición de la variable humedad relativa de aire, la cual contaba con un valor previamente determinado. La activación de las electroválvulas que se encargaba de modificar la humedad se daba al momento que se verificaba una variación en la variable.
  Se realizaron cálculos de ganancias Kp (ganancia proporcional), Ki (ganancia integral) y Kd (ganancia derivativa) para obtener el control del controlador PID (controlador proporcional, integral y derivativo), dichas ganancias fueron introducidas en la programación del Controlador Lógico Programable, obteniendo como resultado final que la regulación de la humedad relativa del

aire en el invernadero dentro de un rango útil del 60 al 70% para el cultivo de tomate aplicado [7].

- En la ciudad de Arequipa, en la "Universidad Nacional de San Agustín de Arequipa" se desarrolló e implemento un sistema de control predictivo para la predicción del comportamiento de ciertas variables climáticas dentro de un invernadero y de esta forma obtener un mejor producto cultivado.
  Dentro de las conclusiones de la investigación se afirmó que aplicando algoritmos para el control predictivo multivariable se podían optimizar el clima del invernadero eficazmente y mejorar el nivel de la producción, además de que existía la posibilidad de implementar un sistema MIMO de dos entradas, es decir, teniendo como entradas el calor de lámparas y flujo de aire, y como salidas la temperatura y la humedad interna, mediante el uso de técnicas de control predictivo [8].

- En una investigación realizada en la ciudad de Trujillo por estudiantes de la "Universidad Nacional de Trujillo" se utilizó una placa Arduino y un aplicativo móvil en Android para poder hacer una maqueta de un invernadero a fin de controlar y monitorizar ciertos parámetros que se habían establecido. La placa de Arduino se utilizó como una tarjeta controladora y receptora de datos y el aplicativo móvil sirvió para monitorizar y controlar el estado del invernadero. Además, se hizo uso del web hosting para obtener un dominio y almacenar la base de datos, con lo recolectado por la placa Arduino y el aplicativo.
  Al automatizar el invernadero se facilitó en maneo de los procesos que influían en el cultivo y la optimización del recurso hídrico. Se logro cumplir con los objetivos de la investigación ya que se obtuvieron respuestas positivas ante las diferentes condiciones climáticas que afectaron al cultivo.
  Se menciono que la aplicación PINVER, la cual fue desarrollada para el control y monitoreo del invernadero facilitó al usuario dichas tareas, ya que podían ser realizadas de manera remota [9].

- En una investigación de la "Universidad Ricardo Palma" en la ciudad de lima se realizó un mini-invernadero automatizado de bajo coste para la producción de rosas, por lo cual se diseñó, simuló e implemento un controlador difuso para el control de las principales variables que intervenían en la automatización del miniinvernadero. Se hizo uso para la simulación el software LabVIEW y para la implementación la plataforma Arduino.
  Se tomaron en cuenta como variables para realizar los lazos de control la humedad relativa, la temperatura, el control del riego, la luminosidad y el fertirriego del cultivo; en base a estas se realizaron algoritmos para su control y la simulación de dichos algoritmos fuer realizada en el software LabVIEW. En la implementación de la solución se albergó un pequeño cultivo de seis plantaciones de

rosas, pudiendo demostrar que era posible el diseño e implementación de un min invernadero automatizado para rosas con un cajo costo de s/836,11 soles [10].

2.1.2. Antecedentes Internacionales

Se encontraron algunos artículos relacionados con el tema de investigación pertenecientes a otros países, los cuales son:

- En artículo publicado en la India titulado "Design of Wireless Sensor Network based Smart Greenhouse System,", se habla sobre el monitoreo de las Smart Greenhouse, mencionan que hicieron uso de WSN (Wireless Sensor Network) ZigBee para realizar el monitoreo del ambiente y control del ambiente, llegando a la conclusión de que el uso de estas tecnologías reducen los costos y aumenta la eficiencia en el campo de cultivo; agregado a ello hicieron uso de un aplicativo móvil para el control remoto de la solución que se había implementado y de esta manera reducir aún más el esfuerzo humano [11].

- Corea es uno de los países que lleva unos años implementando en el desarrollo de sus cultivos las Smart Greenhouse, cada granja con diversos propósitos, algunas para aumentar la productividad, otras para mejorar la calidad de los productos, entre otros. Se hizo un estudio mediante encuestas para conocer el estado real de las Smart Greenhouse en el campo y a su vez clasificarlas en tipos de granjas; llegando a la conclusión de que existen dificultados con respecto al manejo de las Smart Greenhouse por parte de los agricultores cuyas edades eran avanzadas, sin embargo, que si se veían resultados positivos en el uso de estas; aparte de ello clasificaron a las Smart Greenhouse en 7 tipos, yendo desde la más simple hasta la más compleja con respecto a componentes y funcionalidad [12].

- En otro artículo perteneciente a Estados Unidos de la revista "IRJET Journal", se propuso el desarrollo de una Smart Greenhouse, la cual gestionaría información del entorno para crecimiento y desarrollo dentro de los invernaderos de manera remota, para ello recolectaron los datos necesarios para desarrollar una óptima solución que pueda satisfacer las necesidades del agricultor. Implementaron su solución mediante la instalación de sensores que incluyen sensores para suelos, medio ambiente, temperatura y humedad de las hojas y circuito cerrado de televisión; además de ello utilizaron una cámara de supervisión y un GUI (Graphical User Interface) para hacer más intuitiva la solución ante el usuario. Como resultado de su implementación obtuvieron que ninguna operación salió mal y con su propuesta

esperan disminuir los gastos de mano de obra, producir cultivos con mayor calidad y más competentes en el mercado [13].

- En un artículo titulado "IoT based Smart Greenhouse" realizado por investigadores de la India se aplicaron las IoT para demostrar que se pueden producir cultivos libres de insecticidas y pesticidas, creando un clima para el crecimiento adecuado de las plantas e incluso proporcionar una fuente alternativa de ingresos a través de la apicultura, la venta de agua de pozos entubados, etc. Para este proyecto se utilizaron sensores de temperatura, humedad ambiental, y se controlaron los mismos mediante actuadores como un relé para mantener la humedad dentro de valores óptimos, luces led para la iluminación y un sistema de goteo que utilizaba agua de lluvia. Con su trabajo demostraron que se pueden ahorrar costos de agua con la reutilización de la proveniente de la lluvia, además de ello la productividad era mayor debido a la elevación en la tasa de crecimiento de las plantas que utilizaron [14].

- El proyecto "MyGreen" fue desarrollado en Estados Unidos por diversos investigadores los cuales desarrollaron un Invernadero Inteligente habilitado para IoT para la Agricultura Sostenible, el cual consistía en un sistema automatizado para mejorar la productividad en el desarrollo de los cultivos de rosas, con esta investigación se puede afirmar que el sistema brinda información precisa, y que además del aspecto de la productividad del cultivo reduce la carga del trabajo manual mediante la automatización. Al igual que en las demás investigaciones, este proyecto trabajo con sensores y actuadores para las variables climáticas, sin embargo, a diferencia de los otros este automatizo parte de la aplicación de insecticidas y pesticidas. Precisando al final de su investigación que a comparación un trabajo manual, un trabajo de IoT en un comienzo requiere una inversión considerable [15].

2.2. Bases Teórico-Científicas

A continuación, se mencionan los conceptos necesarios para entender en su totalidad el presente proyecto de tesis.

2.2.1. Cultivos de Tomate

2.2.1.1. El Cambio Climático y los Cultivos

El sector agropecuario constituye un gran porcentaje en el territorio peruano un aproximado del 30,1% y cerca de 1,4 millones de personas viven de él, por lo cual el MINAM resalta cuales son los riesgos ambientales que más afectan al territorio peruano que trabaja tanto en el

sector agropecuario como en otros, tales como las heladas, friajes, inundaciones y plagas [3], lo cual se puede apreciar en la Figura I y II.

Figura I: Servicios Ecosistémicos [3].

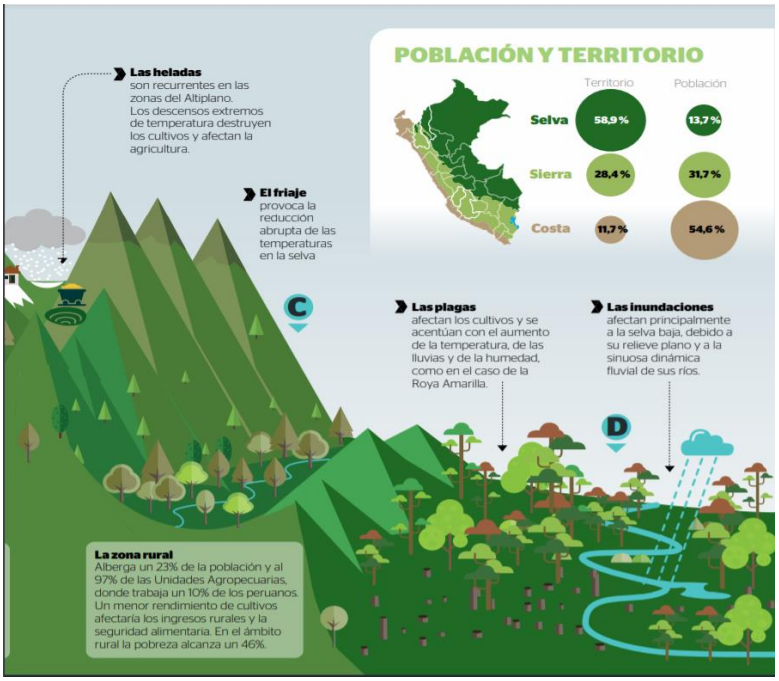


Figura II: Riesgos Ambientales [3].

2.2.1.2. Cultivos de Tomate en la Costa del Perú

El Perú consta de regiones naturales: costa, sierra, selva; siendo la costa la que aporta mayor producción agrícola y en la cual se invierte más para el desarrollo de cultivos.

La costa es una zona árida que abarca aproximadamente 200 000 km2, la cual reúne los suelos más productivos del país, debido a que tiene calidad de tierra, el clima templado, una topografía plana, disponibilidad de agua y el desarrollo que tiene por las inversiones ya antes realizadas.

Uno de los cultivos más comercializados y producidos en la costa del Perú es el tomate, considerándose a este como un producto estrella por la gran demanda que posee en el mercado, siendo en el mercado doméstico la principal fuente de ingreso. Según el ministerio de agricultura y riego, sus principales zonas de producción en el Perú son Lima, La libertad, Ica,

Huaral-Chancay, Barranca, Huacho, Cañete, Arequipa, Lambayeque y Mala. Las cifras presentadas en el año 2016 con respecto a al cultivo de tomates revelo que se produjeron 233,000 toneladas, siendo la mayor parte destinada al consumo interno [4].

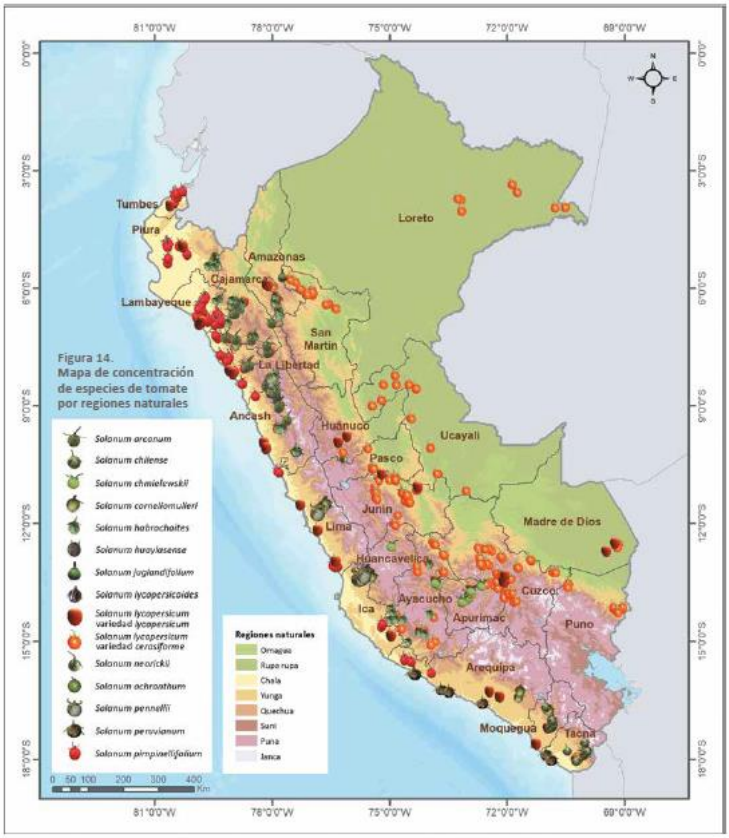


Figura III: Distribución del cultivo de tomate en las diferentes regiones del Perú [2].

2.2.1.3. Tomate Nativo Peruano

En el Perú existen 13 especies de tomate, 12 silvestres y una cultiva, tenido una amplia presencia el tomate nativo (S. lycopersicum variedad cerasiforme) conocida mayormente como tomate cereza o Cherry, encontrándose en los departamentos de Ica, Junín, La libertad, Lambayeque, Lima, Loreto, Madre de Dios, Moquegua, Pasco, Piura, Puno, San Martín, Tacna, Tumbes y Ucayali.
Los encargados de mantener estos cultivos son los agricultores tradicionales, quienes además de realizar la producción de los cultivos, llevan consigo todos los costos de esta, a pesar de ser los más afectados por el cambio climático y sufrir de pobreza extrema e inseguridad alimentaria [1] [2].

Condiciones ambientales necesarias para un óptimo cultivo de tomates [16]:

- Temperatura de la mañana entre los 20°C y 30°C.
- Temperatura de noche entre los 15°C y 17°C.

- Temperatura vegetativa entre los 12°C y 35°C.
- Temperatura para la fecundación entre los 12°C y 25°C.
- Humedad relativa entre el 60% y 80%.
- Nivel de luz alta.

Temperaturas superiores a los 35°C y menores a 12°C pueden ser letales para el cultivo.

Figura IV: Cultivo de tomate en el Perú [2].

2.2.2. Internet on Things (IoT)

IoT son las siglas de Internet of Things, es decir, el internet de las cosas, lo cual describe la red de integración de sensores, software y otras tecnologías con objetos físicos, los cuales reciben y transfieren datos a través de redes inalámbricas con intervención humana mínima [17].

2.2.2.1. Arduino Hardware

Es una gama de productos que incluyen placas, módulos, escudos y kits, para impulsar proyectos de electrónica y IoT; con precios accesibles y una gran variedad de productos. Además de ello Arduino cuenta con su propio IDE, en el cual se trabaja con el leguaje C, para programar principalmente la obtención de los datos de los diversos dispositivos [18] [19].

2.2.2.2. ESP WebSocket Client

Es la implementación del protocolo del cliente WebSocket el cual permite que exista una conexión bidireccional entre un cliente y un host que ha optado por recibir lo que el cliente está enviando [20].

2.2.2.3. Sensores

Un sensor es un dispositivo convierte un fenómeno físico en un voltaje analógico o digital, al detectar algún cambio en el entorno, respondiendo con alguna salida en el otro sistema, es decir, un valor legible para el ser humano. Algunos de los sensores más comunes para conocer los valores de las variables ambientales que existen son los sensores de temperatura y humedad, los sensores de luminosidad, los sensores de peso, entre otros [21].

2.2.2.4. Actuadores

Un actuador es un dispositivo que tiene la capacidad de energía hidráulica, neumática o eléctrica en la activación de un proceso, con el fin de generar un efecto sobre elemento externo, sin embargo, en los proyectos de IoT se suele trabajar con los actuadores de tipo eléctrico [22].

2.2.2.5. Smart Greenhouse

Una Smart Greenhouse es una revolución en la agricultura, creando un microclima autorregulado y adecuado para el crecimiento de las plantas mediante el uso de sensores, actuadores con sistemas de monitoreo y control que optimizan las condiciones de crecimiento y automatizan el proceso de cultivo [23].
Al integrar el IoT con otras tecnologías permite que la interacción humana en los cultivos se reduzca y que de esta manera se pueda ahorrar en costos de personal, mejorando el rendimiento mediante el control y monitoreo remoto de los cultivos, reduciendo los impactos del cambio climático [24].
Las posibles soluciones basadas en IoT para los desafíos agrícolas modernos se centrarían en torno la aplicación de sistemas IoT en agricultura inteligente e invernaderos abarcando diversos tipos de tecnologías [25] [26] [27] [28] [29].

2.2.3. Aplicación Móvil

Son programas diseñados para ser ejecutados en dispositivos móviles como tabletas, celulares y otros. Ofreciendo la posibilidad de acceder a servicios, información y otros desde cualquier lugar [30].

2.2.3.1. Aplicación Híbrida

Estas aplicaciones usan funcionalidades nativas de los teléfonos, pero el mismo código sirve para varios sistemas operativos. Tienen un coste de desarrollo menor al de una aplicación nativa. Además, permiten el uso de todas las funcionalidades que nos ofrecen los dispositivos como son: contactos, cámara, GPS, etc. [30]
A diferencia de las aplicaciones nativas, se desarrollan con lenguajes propios de las webabpp, es decir, HTML, Javascript y CSS por lo que permite su uso en diferentes plataformas.

2.2.3.2. Plataformas de Desarrollo

Las plataformas que se utilizaron para desarrollar el producto acreditable son las siguientes:

- **Android**

  Es un sistema operativo de código abierto para dispositivos móviles, siendo más populares en el mercado, que les permite a los usuarios controlar aún más sus herramientas y ajustes de seguridad [31].

- **Arduino IDE**

  Es una gama de productos que incluyen placas, módulos, escudos y kits, para impulsar proyectos de electrónica y IoT; con precios accesibles y una gran variedad de productos. Además de ello cuenta con un IDE el cual se trabaja en el lenguaje C [18] [19].

- **ReactJs**

  Es un framework desarrollado por Facebook, siendo una biblioteca JavaScript de código abierto muy extendida. puede utilizar para crear todo tipo de aplicaciones web, para móviles, e interfaces y mucho más. ReactJS es un framework JavaScript moderno, declarativo y eficiente que permite un desarrollo flexible y hace que la creación de interfaces de usuario interactivas/front-end sea divertida y completamente indolora [32].

- **Visual Studio Code**

  Es un editor de texto de código abierto que soporta diversos lenguajes de programación, permitiendo gestionar tus propios atajos de teclado y refactorizar el código. Además de ello cuenta con muchas extensiones que hacen el trabajo del programador más sencillo [33].

Algunos de los lenguajes con los que se puede trabajar el Visual Studio Code son: Python, C / C++, JavaScript, Dart, etc.

2.2.3.3. Bases de Datos

- **Base de datos Relacional**

  También conocida como SQL, es aquella que usa el ya conocido esquema de filas y columnas; en donde cada fila de la tabla es un registro único llamado clave y las columnas contienen atributos de los datos [34].

- **JSON**

  Cuando se habla de los datos JSON se hace referencia a una notación de objetos de JavaScript que almacena datos en un formato jerárquico y los representan como pares clave/valor en un formato semi estructurado. Se utiliza habitualmente ya que ofrece flexibilidad y compatibilidad debido al formato que utiliza, los archivos JSON a menudo son enviados por la web por diversos dispositivos incluyendo los de IoT para comunicaciones unirelacionales, birrelacionales o por aplicaciones de cliente [34], el formato JSON permite que se pueden estructurar los datos en forma de texto y sean enviados de manera más sencilla a la base de datos o directamente a la aplicación que se está desarrollando.

- **Graphql**

  Es un lenguaje de consulta y manipulación de datos para APIs, el cual soporta la lectura, escritura y suscripción de cambios de información en tiempo real, este último hace referencia a que cada vez que se realice un cambio la consulta se volverá a ejecutar. Además, está disponible para múltiples lenguajes como: JavaScript, Perl, Python, Ruby, Java, C++,11 C#, PHP, R, entre otros [35].

- **Docker**

  Es un proyecto de código abierto lanzado en marzo del 2013, el cual trabaja mediante contenedores, Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de forma rápida y confiable de un entorno informático a otro [36].

III. MATERIALES Y MÉTODOS

3.1. Tipo de Investigación

Para el desarrollo de esta investigación, se ha optado por realizar una investigación de tipo experimental, debido a que se utilizarán las tecnologías y conocimientos existentes para desarrollar una solución IoT para especies de tomate nativos del Perú, para lo cual no se han encontrado antecedentes que aplique específicamente una solución de este tipo de cultivo. Para ello se hará una comparación tomando en cuenta que se tendrán dos grupos:

- Grupo Experimental: Cultivo A
- Grupo Control: Cultivo B

Tomando en cuenta que el grupo experimental será el cultivo en cual se hará uso de la solución desarrollada y el grupo control, será aquel tendrá un monitoreo constante, pero sin hacer uso de la solución planteada.

3.2. Métodos de investigación

Para la realización de la investigación, se cuenta con los siguientes métodos:

Tabla I: Métodos de investigación

Método	Descripción
Analítico	Se identificarán y analizaran las variables que interviene en el ambiente en el cultivo de tomates en la costa del Perú
Deductivo	Se diseñará la solución de Smart Green House utilizando las IoT, juntamente con un aplicativo móvil que permitirá mejorar la experiencia del usuario
Implementación	Desarrollar e implementar la solución propuesta para evaluar su funcionamiento

3.3. Técnicas e instrumentos de recolección de datos

Para la recolección de datos de la investigación, se cuenta con las siguientes técnicas e instrumentos:

Tabla II: Técnicas e instrumentos de recolección de datos

Técnicas	Instrumentos	Elementos de la población	Propósito
Entrevista	Cuestionario	Ingeniero agrónomo/ Cultivador	Conocer más acerca del cultivo de tomates en el Perú.
Análisis documental	Base de datos de investigación	Documentos	Obtener más información de la realidad problemática. Conocer soluciones similares aplicadas a cultivos.
Observación	Ficha de registro de datos	Al cultivo de tomates nativo que se está realizando con la solución	Identificar las diferencias entre el crecimiento de los tomates con la solución y sin ella

3.4. Procedimientos

3.4.1. Metodología de Desarrollo

Para el desarrollo de software existen dos tipos de metodologías las cuales son metodologías ágiles y metodologías tradicionales, siendo las ágiles las más actuales y usadas en la actualidad debido a las facilidades que proveen al momento de trabajar en equipo, la planificación, progreso mediante entregas incrementales y aprendizaje continuo en cada una de sus etapas, a diferencia de estas las metodologías tradicionales, las cuales se centran en llegar a los objetivos de cada fase.

Para la selección de la metodología se tomó en cuenta utilizar una metodología ágil, debido a que las tradicionales no se adaptan al desarrollo de esta solución a pesar de lo fácil que es comprender como utilizarlas; la popularidad y resultados obtenidos aplicando las metodologías ágiles son mejores. Por ello se escogió entre la metodología XP y la Scrum, ambas siendo muy similares en características, sin embargo, la metodología XP trabaja promoviendo la programación en pares, estándares de código, espacio de trabajo informativo, marcha sostenible e integración continua, a pesar de que todas estas son prácticas positivas, se considera a esta metodología muy estricta con respecto a los cambio, por lo contrario, Scrum es más flexible en ese aspecto, sugiriendo diferentes artefactos que permiten llevar la trazabilidad del proyecto a través del tiempo, como Pila del producto o

Product Backlog, Pila del sprint o sprint backlog y Gráfico de avance o Burndown Chart [37]. Por lo tanto, se elegio a Scrum como la metodología a utilizar, debido a que permite que el proyecto se lleve a cabo tomando en cuenta la posibilidad de realizar cambios habiendo terminado ya una iteración, no obstante, advierte a los desarrolladores sean conscientes de los cambios realizados. Además, la metodología Scrum no solo es utilizada para desarrollar aplicaciones móviles, sino también para desarrollar sistemas con IoT ya que es una metodología que trabaja para producir un sistema en un entorno en constante cambio de forma flexible y funciona en cualquier dominio [38].
A continuación, describiremos el primer Sprint, el cual es la primera etapa de la metodología Scrum, en la cual definimos las bases del proyecto, es decir los requerimientos del negocio y los pasos a seguir durante el desarrollo del proyecto.

3.4.1.1. Definición del Proyecto

El presente proyecto tendrá como finalidad desarrollar una Smart Green House, haciendo uso de las IoT y un aplicativo móvil para que pueda contribuir a mejorar los cultivos de tomate bajo techo, proporcionado mayor control y monitoreo en el ambiente del cultivo, mostrando los datos obtenidos por los sensores y actuadores en un aplicativo móvil, facilitándole así el trabajo a los agricultores y disminuyendo las perdidas en producción, se tiene establecido como tiempo estimado un promedio de 220 días para la finalización del proyecto.

3.4.1.2. Determinación del Producto Final

El presente proyecto finalizará cuando el Smart Green House basado en IoT cuente con la implementación del circuito con los sensores y actuadores seleccionados, permitiendo la obtención de los datos de las principales variables identificadas para el cultivo, así como el accionar de los actuadores para que dichas variables se puedan mantener dentro de los parámetros establecidos por el usuario mediante la aplicación móvil desarrollada permitiendo la visualización de los datos que se manejan en el circuito (sensores y actuadores) y el registro de información asociada al cultivo , siendo aplicado en un cultivo de tomates nativos del Perú.

3.4.1.3. Definición del Product Backlog Inicial

Tabla III: Product Backlog del proyecto

ITEM	PRODUCT BACKLOG
1	Implementar el circuito con los componentes IoT en la greenhouse
2	Desarrollar la base de datos
3	Gestionar el control y monitoreo del ambiente
4	Gestionar Registros
5	Implementar recordatorios de eventos
6	Implementar reportes

Priorizando el product backlog

El Product Owner es también el encargado de priorizar los requerimientos del Product Backlog, de acuerdo con su funcionalidad dentro la implementación.

Tabla IV: Priorización del Product Backlog

ITEM	PRIORIDAD	PRODUCT BACKLOG
1	1	Implementar el circuito con los componentes IoT en la greenhouse
1	2	Desarrollar la base de datos
3	3	Gestionar el control y monitoreo del ambiente
4	4	Gestionar Registros
5	5	Implementar recordatorios de eventos
6	6	Implementar reportes

Identificando complejidad

En esta tarea se califica la complejidad de las tareas del Product backlog en una escala del 1 al 5.

Tabla V: Complejidad del Product Backlog

ITEM	PRIORIDAD	COMPLEJIDAD	PRODUCT BACKLOG
1	1	5	Implementar el circuito con los componentes IoT en la greenhouse
2	2	3	Desarrollar la base de datos
3	3	3	Gestionar el control y monitoreo del ambiente
4	4	4	Gestionar Registros
5	5	2	Implementar recordatorios de eventos
6	6	3	Implementar reportes

Asignando un valor en story points

Para asignar los puntos de historia de uso el equipo utilizó el método "Planning Poker" y se asignó un valor a la historia de usuario de menor complejidad, luego de ello se colocará el valor de los demás ítems, tomando como base el primero. Las cartas utilizadas para este método son las siguientes:

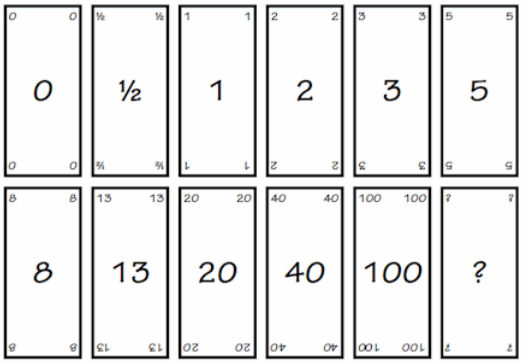


Figura V: Planning Poker [39]

Tabla VI: Puntuación del Product Backlog

ITEM	PRIORIDAD	COMPLEJIDAD	PUNTOS	PRODUCT BACKLOG
1	1	5	13	Implementar el circuito con los componentes IoT en la greenhouse
2	2	3	5	Desarrollar la base de datos
3	3	3	5	Gestionar el control y monitoreo del ambiente
4	4	4	8	Gestionar Registros
5	5	2	3	Implementar recordatorios de eventos
6	6	3	5	Implementar reportes

3.4.1.4. Descripción de los Sprints e Historias de Usuarios

Tabla VII: Product Backlog del proyecto

Sprint	Historias de Usuario	Tareas
IMPLEMENTAR EL CIRCUITO CON LOS COMPONENTES IOT EN LA GREENHOUSE	Implementación del circuito con los componentes IoT en la greenhouse	Diseño de la Greenhouse
		Construcción de la Greenhouse
		Identificar los componentes necesarios para el circuito
		Evaluar y seleccionar los componentes a utilizar en el circuito
		Diseño del circuito con los componentes IoT
		Prototipo del circuito
		Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores
		Captura de los datos de los sensores
		Implementar el circuito diseñado en la Greenhouse

DESARROLLAR LA BASE DE DATOS	Desarrollo de la Base de Datos	Diseño del modelo entidad relación
		Configuración del Servidor de Base de datos
		Configuración del ESP32 WebSocket Client
GESTIONAR EL CONTROL Y MONITOREO DEL AMBIENTE	Gestión del Control y monitoreo del ambiente	Diseñar la interfaz de home
		Desarrollo de las funcionalidades de la interfaz home
		Diseño de la interfaz de actuadores
		Desarrollo de las funcionalidades de la interfaz actuadores
GESTIONAR REGISTROS	Gestión de la Planta, sus Fases y Parámetros	Diseño de la interfaz de usuario de listar plantas
		Diseño de la interfaz de usuario de registro y modificación de planta
		Diseño de la interfaz de usuario de listar fases de una planta
		Diseño de la interfaz de usuario de registro y modificación de la fase de una planta
		Diseño de la interfaz de usuario de listar parámetros de una planta
		Diseño de la interfaz de usuario de registro y modificación de los parámetros de una planta
		Desarrollo de la funcionalidad de gestionar planta
		Desarrollo de la funcionalidad de gestionar fases de una planta
		Desarrollo de la funcionalidad de gestionar los parámetros de una planta
	Gestión del cultivo	Diseño de la interfaz de usuario de listar cultivos

		Diseño de la interfaz de usuario de registro y modificación de un cultivo
		Desarrollo de la funcionalidad de gestionar cultivo
	Gestión de la cosecha	Diseño de la interfaz de usuario de listar cosechas
		Diseño de la interfaz de usuario de registro y modificación de una cosecha
		Desarrollo de la funcionalidad de gestionar cosecha
IMPLEMENTAR RECORDATORIOS DE EVENTOS	Implementación de recordatorios de eventos	Diseño de la interfaz de usuario de listar recordatorios de eventos
		Diseño de la interfaz de usuario de registro y modificación de los recordatorios de eventos
		Desarrollo de la funcionalidad de los recordatorios de eventos
IMPLEMENTAR REPORTES	Implementación de reportes	Diseño de la interfaz de reportes
		Desarrollo de la funcionalidad de los reportes

3.4.2. Producto Acreditable

3.4.2.1. Interfaces

Figura VI: Interfaz de Menú Principal

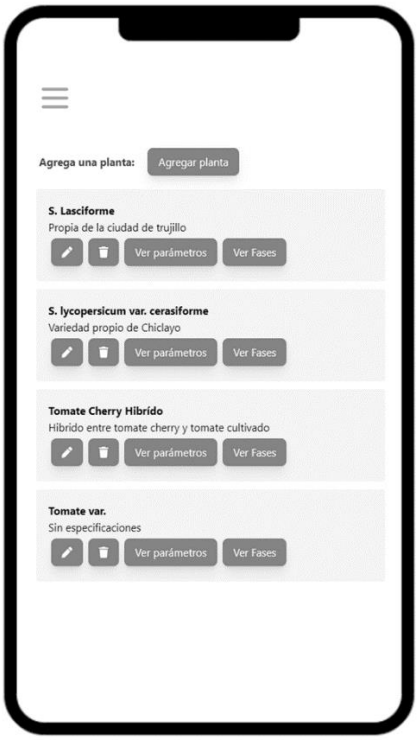


Figura VII: Interfaz de Planta

Figura VIII: Interfaz de Agregar o Modificar Planta

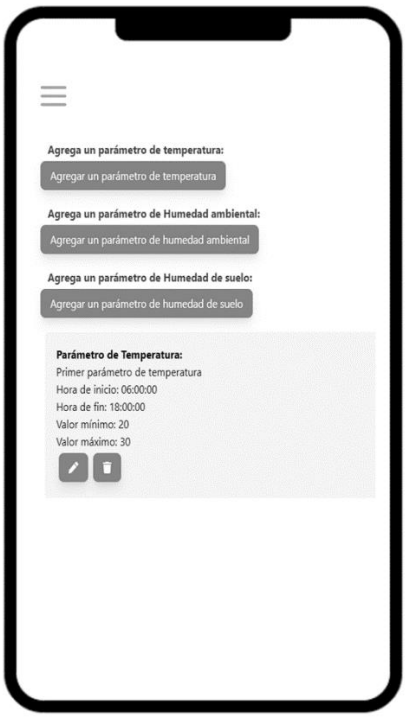


Figura IX: Interfaz del Parámetro de Planta

Figura X: Interfaz de Agregar o Modificar Parámetro de Temperatura de Planta

Figura XI: Interfaz de Agregar o Modificar Parámetro de Humedad Ambiental de Planta

Figura XII: Interfaz de Agregar o Modificar Parámetro de Humedad de suelo de Planta

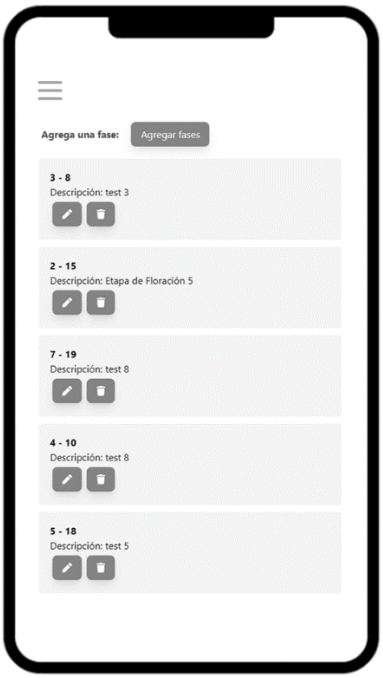


Figura XIII: Interfaz de las Fases de Planta

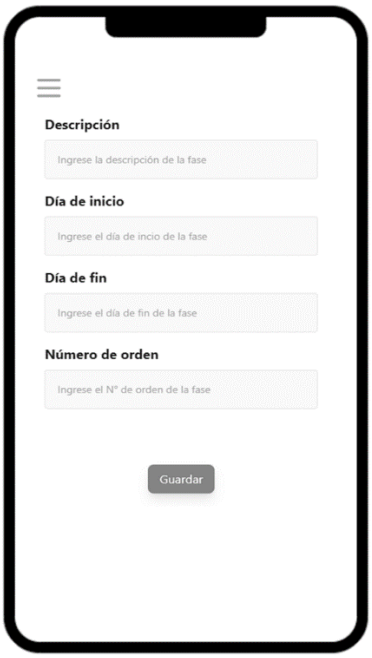


Figura XIV: Interfaz de Agregar o Modificar una Fase de Planta

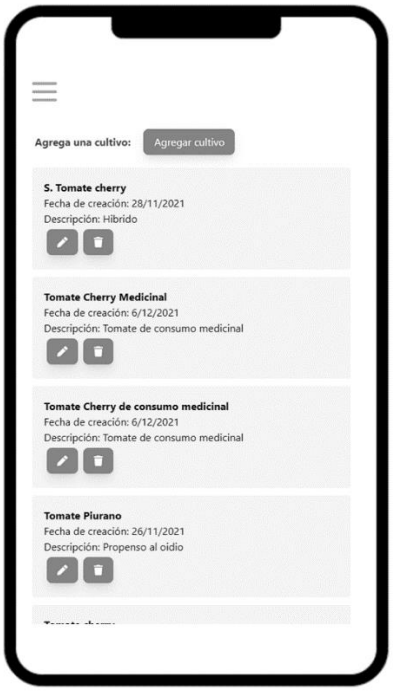


Figura XV: Interfaz de Cultivo

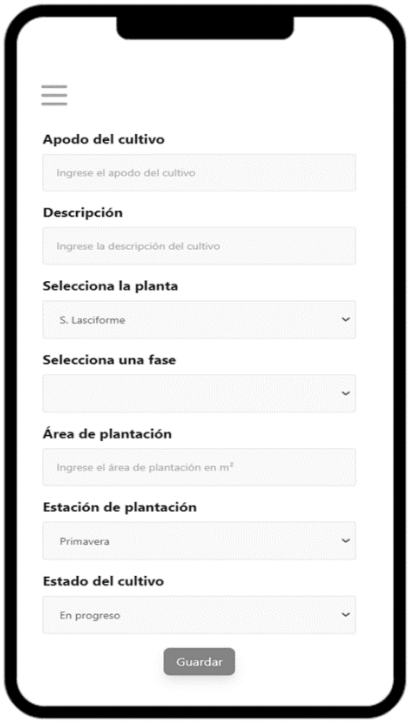


Figura XVI: Interfaz de Agregar o Modificar Cultivo

Figura XVI: Interfaz de Cosecha

Figura XVIII: Interfaz de Agregar o Modificar Cosecha

Figura XIX: Interfaz de Recordatorio

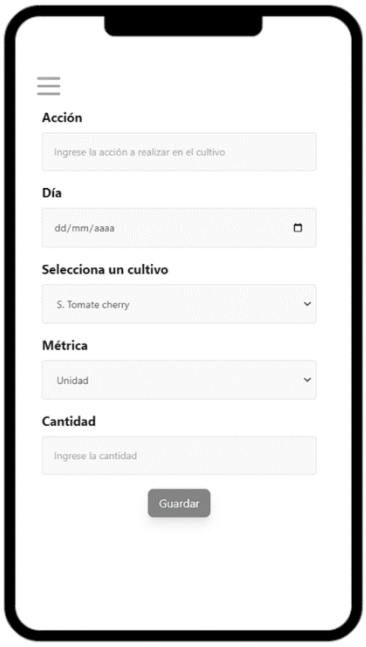


Figura XX: Interfaz de Agregar o Modificar Recordatorio

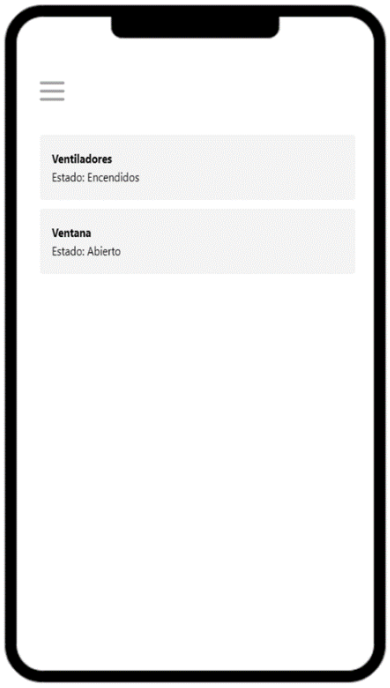


Figura XXI: Interfaz de Actuadores

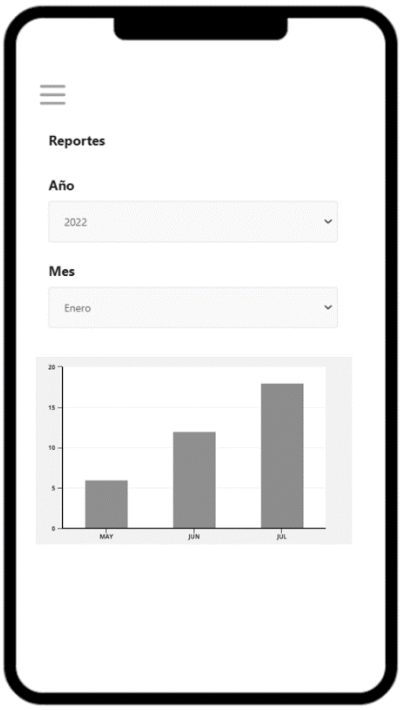


Figura XXII: Interfaz de Reportes

3.4.2.2. Arquitectura

La arquitectura considerada para la implementación del producto acreditable es la siguiente:

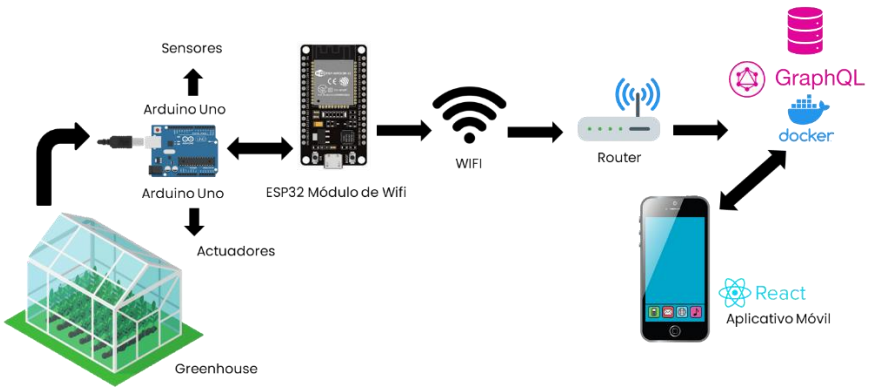


Figura XXIII: Arquitectura de la solución

Como se puede apreciar en la ***Figura XXII: Arquitectura de la solución***, hacia la izquierda tenemos a la greenhouse o invernadero, dentro de la cual se ubica al cultivo de tomates, en este invernadero se posicionaron los diversos componentes IoT de la siguiente manera: el sensor de temperatura y humedad ambiental ubicado lo más cerca del cultivo, el sensor de humedad de suelo conectado al suelo donde se encuentran cultivados, el sensor de luz en la zona en la cual le cae la iluminación al cultivo y el sensor de lluvia cerca a la ventana para que pueda detectar la lluvia. Por otro lado, los actuadores que se han contemplado son 2, un servo para la apertura y cierre de la ventana y los ventiladores los cuales sen ubicaran muy cerca al cultivo.
Para poder gestionar los datos de los sensores se hará uso de la placa de Arduino, la cual permitirá la recepción de los datos, posterior a ello se conectará con el módulo de WIFI ESP32 el cual enviará dichos datos al servidor para el cual se hizo uso de las tecnologías de Graphql y Docker; el servidor no solo se hará cargo de hacer llegar dichos datos a la aplicación, sino también tomará la decisión en base a los datos recibidos de cuáles son las acciones que deben realizar los actuadores y se las hará llegar a la placa de Arduino mediante el módulo ESP32.
Los datos que recepciona el servidor, así como las ordenes que envía, llegaran también al aplicativo para que sea mostrado al usuario. Cabe resaltar que el aplicativo mencionado fue desarrollado con el uso del Framework de React.

3.4.2.3. Infraestructura Tecnológica

Considerando la arquitectura anteriormente descrita, se definen las características de cada uno de sus componentes.
Primeramente, describiremos los componentes IoT, los cuales son los siguientes:

- **ESP32-WROOM-32:** El módulo de WIFI ESP32, pertenece a la gama de los productos de Espressif. Se conecta de manera serial al Arduino por los pines RX y TX. Siendo el intermediario entre la placa de Arduino y el servidor.
- **ARDUINO_UNO_R3:** La placa de Arduino Uno, es una de las más utilizadas en el mercado debido a la gran cantidad de puertos que posee y sus diversas utilizades en el campo del IoT, en este caso se encontró conectada a los diversos sensores y actuadores, así como al módulo de WIFI ESP32 de manera serial por los pines RX y TX.
- **SENSOR_FC28:** El componente con este nombre hace referencia a los sensores de humedad de suelo el cual se conectó a la placa Arduino mediante los pines analógicos.
- **SENSOR DE LLUVIA Y NIEVE FC-27:** El componente con este nombre hace referencia al sensor de lluvia, el cual también se

encuentra conectado a la placa Arduino mediante uno de los pines analógicos.

- **SENSOR DE TEMPERATURA Y HUMEDAD RELATIVA SHT31X:** El componente con este nombre hace referencia al sensor de temperatura y humedad relativa, el cual se encuentra conectado mediante el protocolo de conexión serial I2C a uno de los pines analógicos de la placa Arduino.
- **MÓDULO BH1750:** El componente con este nombre hace referencia al sensor de luminosidad, el cual de igual forma que el SHT31X se encuentra conectado mediante el protocolo de conexión serial I2C a uno de los pines analógicos de la placa Arduino.
- **MOTOR_SERVO:** El componente con este nombre hace referencia al motor servo de 11kg seleccionado, el cual se utilizó para la apertura y cierre de la ventana, este se encuentra conectado a la placa Arduino mediante uno de los pines digitales.
- **VENTILADORES:** Los componentes con este nombre hacen referencia a los ventiladores utilizados para ventilar el cultivo, en este caso estos ventiladores utilizan un voltaje de entre 6 y 12 voltios, estos se encuentran conectados a la placa Arduino mediante los pines digitales.
- **RESISTENCIA:** El componente con este nombre hace referencia a las resistencias utilizadas para los ventiladores, para que así el flujo de corriente no dañe a los ventiladores, el valor de estas es de 220.

Con respecto a las tecnologías utilizadas para el desarrollo del aplicativo:

- **ARDUINO IDE:** La programación para la obtención de los datos de los sensores y acciones de los actuadores se hizo uso del IDE de Arduino, trabajando con el monitor serie que permitía ver la recepción de datos y envió de acciones previamente a la conexión de la placa con el módulo ESP32, el servidor y el aplicativo.
- **GRAPHQL Y DOCKER**: Se trabajo con un servidor local el cual se desarrolló con Graphql para la suscripción de cambios de información en tiempo real, así como la realización de consultas y Docker para una ejecución más rápida del aplicativo.
- **REACT:** El aplicativo se desarrolló solo para ser utilizado por dispositivos Android, debido al uso de la tecnología de React también podría ser utilizado como aplicativo móvil.

3.4.3. Manual de Usuario

A continuación, se muestra el manual de usuario con las principales interfaces, este manual se elaboró con la finalidad de ayudar a los usuarios en el uso del aplicativo desarrollado.

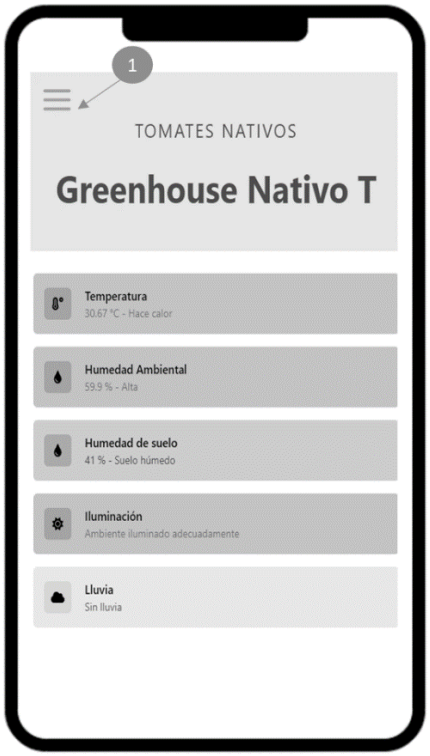


Figura XXIV: Interfaz Detallada Menú Principal

Como se puede apreciar en la Figura XXIV: Interfaz Detallada Menú Principal se muestran los valores de los diversos datos obtenidos por los sensores, y en la parte superior izquierda señalado con el número uno se encuentra el botón del despliegue del menú para acceder a las otras funcionalidades del sistema.

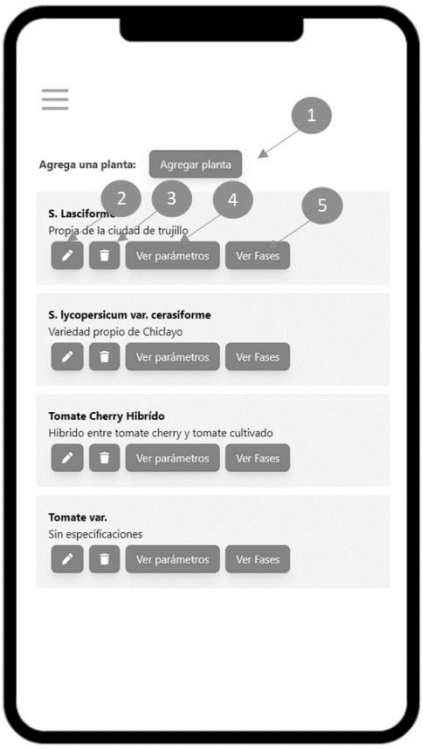


Figura XXV: Interfaz Detallada de Planta

En la Figura XXV: Interfaz Detallada Planta se la lista de las plantas registradas, los significados de cada uno de los botones señalados son los siguientes:

- Botón 1: Botón de “agregar planta”, el cual al ser presionado te lleva al formulario para poder agregar una nueva planta
- Botón 2: Botón de “editar planta”, el cual permite la modificación de uno o más datos de la planta señalada.
- Botón 3: Botón “eliminar”, el cual permite eliminar una planta.
- Botón 4: Botón “ver parámetros”, el cual al ser presionado te lleva a la interfaz donde se encuentran los parámetros registrados para una planta y te permite agregar otros parámetros.
- Botón 5: Botón “ver fases”, al ser presionado este botón te lleva a una interfaz que muestra las fases registradas y permite agregar más de estas.

Figura XXVI: Interfaz Detallada del Parámetro de Planta

En la Figura XXVI: Interfaz Detallada del Parámetro de Planta se la lista de los parámetros de las plantas registradas, los significados de cada uno de los botones señalados son los siguientes:

- Botón 1: Botón de “agregar un parámetro de temperatura”, el cual al ser presionado te lleva al formulario para poder agregar los valores para el parámetro de la temperatura.
- Botón 2: Botón “agregar un parámetro de humedad ambiental”, el cual al ser presionado te lleva al formulario para poder agregar los valores para el parámetro de la humedad ambiental, solo se puede agregar un parámetro de humedad ambiental.
- Botón 3: Botón “agregar un parámetro de humedad de suelo”, el cual al ser presionado te lleva al formulario para poder agregar los valores para el parámetro de suelo, solo se puede agregar un parámetro de humedad de suelo.
- Botón 4: Botón “editar”, el cual al ser presionado permite cambiar uno o más datos del parámetro.
- Botón 5: Botón “eliminar”, al ser presionado este botón elimina el parámetro.

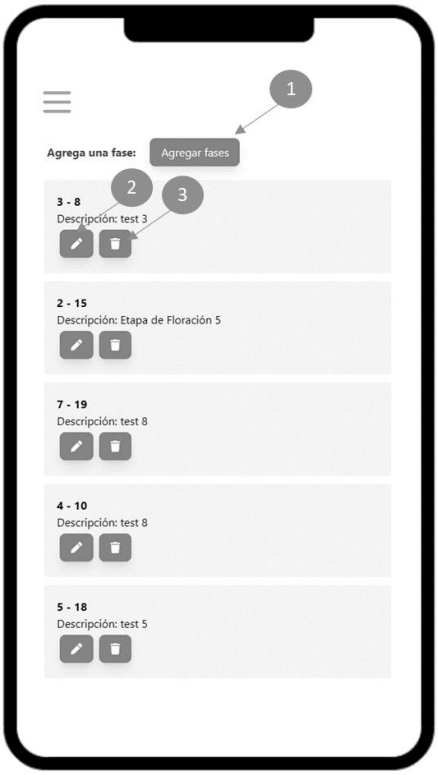


Figura XXVII: Interfaz Detallada de Fases

En la Figura XXVII: Interfaz Detallada de Fases se la lista de las fases de las plantas registradas, los significados de cada uno de los botones señalados son los siguientes:

- Botón 1: Botón de “agregar fase”, el cual al ser presionado te lleva al formulario para poder agregar la fase de una planta.
- Botón 2: Botón “editar”, el cual al ser presionado te permite cambiar uno o más datos de una fase.
- Botón 3: Botón “eliminar”, el cual al ser presionado elimina la fase seleccionada.

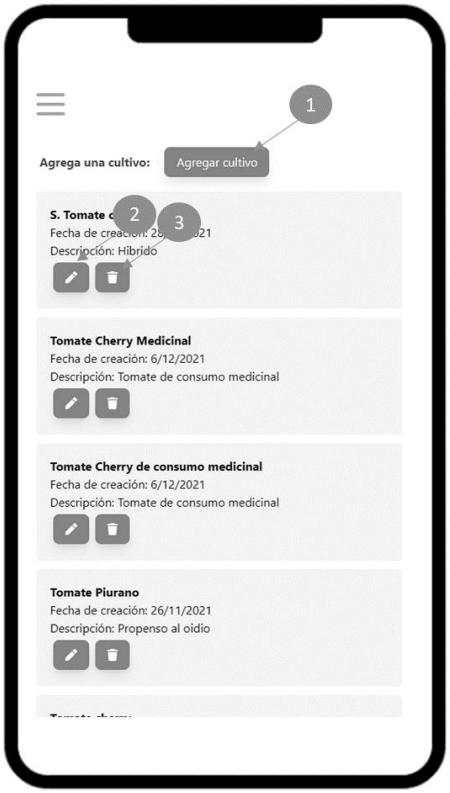


Figura XXVIII: Interfaz Detallada de Cultivos

En la Figura XXVIII: Interfaz Detallada de Cultivos se la lista de los cultivos registrados, los significados de cada uno de los botones señalados son los siguientes:

- Botón 1: Botón de “agregar cultivo”, el cual al ser presionado te lleva al formulario para poder agregar un cultivo.
- Botón 2: Botón “editar”, el cual al ser presionado te permite cambiar uno o más datos de un cultivo.
- Botón 3: Botón “eliminar”, el cual al ser presionado elimina el cultivo seleccionado.

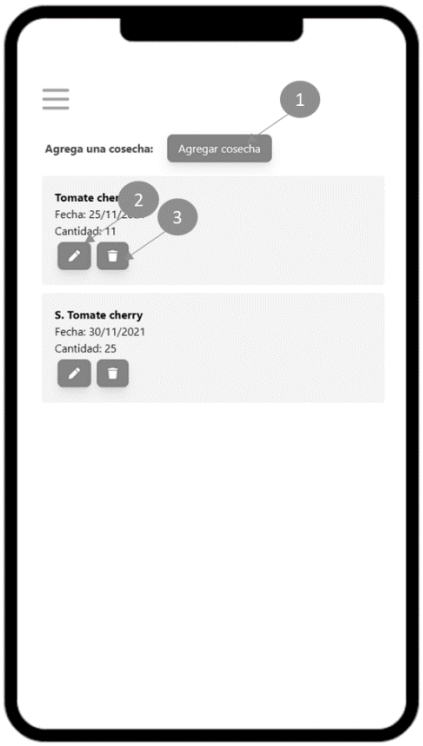


Figura XXIX: Interfaz Detallada de Cosecha

En la Figura XXIX: Interfaz Detallada de Cosecha se la lista de las cosechas registradas, los significados de cada uno de los botones señalados son los siguientes:

- Botón 1: Botón de “agregar cosecha”, el cual al ser presionado te lleva al formulario para poder agregar una cosecha.
- Botón 2: Botón “editar”, el cual al ser presionado te permite cambiar uno o más datos de una cosecha.
- Botón 3: Botón “eliminar”, el cual al ser presionado elimina la cosecha seleccionada.

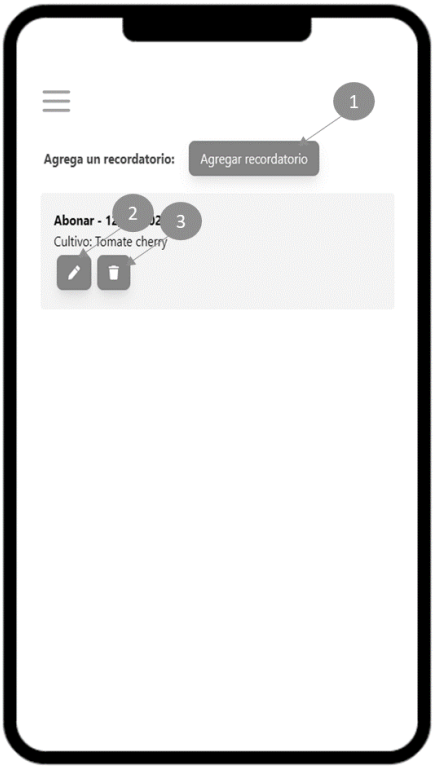


Figura XXX: Interfaz Detallada de Recordatorio

En la Figura XXX: Interfaz Detallada de Recordatorio se la lista de los recordatorios registrados, los significados de cada uno de los botones señalados son los siguientes:

- Botón 1: Botón de “agregar recordatorio”, el cual al ser presionado te lleva al formulario para poder agregar un recordatorio.
- Botón 2: Botón “editar”, el cual al ser presionado te permite cambiar uno o más datos de un recordatorio.
- Botón 3: Botón “eliminar”, el cual al ser presionado elimina el recordatorio seleccionado.

3.5. Matriz de Consistencia

Tabla VIII: Matriz de consistencia

<table>
<tr><td colspan="2">FORMULACIÓN DEL PROBLEMA</td><td colspan="4">MÉTODOLOGÍA DE INVESTIGACIÓN</td></tr>
<tr><td colspan="2">¿Cómo se podría controlar y monitorear el ambiente en cultivos de tomates nativos en tiempo real en la región de Lambayeque para contribuir a su conservación?</td><td colspan="4">TIPO DE INVESTIGACIÓN
Experimental</td></tr>
<tr><td colspan="2">OBJETIVO GENERAL</td><td>MÉTODO</td><td colspan="3">DESCRIPCIÓN</td></tr>
<tr><td colspan="2" rowspan="4">Desarrollar una Smart Greenhouse para la conservación de cultivos de tomates nativos en la región Lambayeque.</td><td>Analítico</td><td colspan="3">Se identificarán y analizaran las variables que interviene en el ambiente en el cultivo de tomates en la costa del Perú</td></tr>
<tr><td>Deductivo</td><td colspan="3">Se diseñará la solución de Smart Green House utilizando las IoT, juntamente con un aplicativo móvil que permitirá mejorar la experiencia del usuario</td></tr>
<tr><td>Implementación</td><td colspan="3">Desarrollar e implementar la solución propuesta para evaluar su funcionamiento</td></tr>
<tr><td>TÉCNICAS</td><td>INSTRUMENTOS</td><td>ELEMENTOS DE LA POBLACIÓN</td><td>PROPÓSITO</td></tr>
<tr><td colspan="2" rowspan="3"></td><td>Entrevista</td><td>Cuestionario</td><td>Ingeniero agrónomo</td><td>Conocer mejor la realidad problemática y si existe la necesidad de la solución propuesta.</td></tr>
<tr><td>Entrevista</td><td>Cuestionario</td><td>Usuario que probará la solución</td><td>Medir el nivel de confianza que tiene el usuario con los resultados obtenidos</td></tr>
<tr><td>Observación</td><td>Ficha de seguimiento</td><td>Al cultivo de tomates nativo que se está realizando con la solución</td><td>Identificar las diferencias entre el crecimiento de los tomates con la solución y sin ella</td></tr>
<tr><td>OBJETIVOS ESPECÍFICOS</td><td colspan="3">DESCRIPCIÓN DEL LOGRO DE LOS OBJETIVOS ESPECÍFICOS</td><td>INDICADORES</td></tr>
</table>

Definir los componentes eléctricos y electrónicos a utilizar en el circuito de la Smart Greenhouse.	Se busca identificar los componentes eléctricos y electrónicos que cumplan con la rúbrica establecida.	Componentes que cumplan con mayor puntaje los criterios de selección.
Implementar una aplicación móvil para el control del ambiente y monitoreo en el cultivo de tomates nativos en la Smart Greenhouse.	Se desea que la aplicación registre los datos que los sensores reciben y los datos que los actuadores emitirán.	Pruebas de caja blanca Pruebas de caja negra Escala de funcionalidad
Validar la implementación del Smart greenhouse para la determinación de su efectividad en el proceso óptimo del cultivo de las plantas de tomate nativo.	Se desea alcanzar un alto grado mejoría en el desarrollo de cultivos de tomates nativos.	Indicadores de rendimiento para el cultivo

3.6. Consideraciones Éticas

En el caso de esta investigación no se trabajaron con animales o personas directamente, pero si con la asesoría de un Ingeniero Agrónomo, aunque no se hizo uso de su información personal.
Con respecto al lugar donde se desarrolló la investigación se respetó la discreción de la ubicación, así como de los propietarios.

IV. RESULTADOS Y DISCUSIÓN

4.1. En Base a la Metodología Utilizada

4.1.1. Sprint 1: Implementar el circuito con los componentes IoT en la Greenhouse

A continuación, se presentarán y describirán las tareas correspondientes a la construcción de la parte IoT del proyecto.

Tabla IX: Identificar los componentes necesarios para el circuito

Componentes para el circuito		
Módulo de comunicación y placas		
Componente	**Descripción**	**Seleccionado**
Módulo de WIFI	Permite él envió de datos mediante WIFI	Si
Módulo de Bluetooth	Permite él envió de datos mediante Bluetooth	No
Placa basada en microcontroladores	Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, estos se encuentran en la placa permitiendo recepcionar y enviar datos digital y analógicamente mediante los diversos puertos que posee.	Si
Sensores		

Componente	Descripción	Seleccionado
Sensor de temperatura	Capta la temperatura del ambiente en el cual se encuentra ubicado, enviando un valor lo más preciso posible dependiendo de la precisión y rango que posea.	Si
Sensor de humedad	Capta la humedad del ambiente en el cual se encuentra ubicado, enviando un valor lo más preciso posible dependiendo de la precisión y rango que posea.	Si
Sensor de humedad de suelo	Capta la humedad del suelo o tierra en el cual se encuentra ubicado, enviando un valor lo más preciso posible dependiendo de la precisión y rango que posea.	Si
Sensor de lluvia	Permite saber si en el ambiente está lloviendo.	Si
Sensor de Co2	Permite conocer la existencia de CO2	No
Sensor de luz	Permite conocer el nivel de luz que llega al lugar en el que se encuentra ubicado.	Si
Actuadores		
Servo motores	Es un actuador rotativo el cual permite un control preciso en términos de posición angular, aceleración y velocidad.	Si
Ventiladores	Es un actuador que permite la ventilación de un determinado ambiente.	Si
Humidificadores	Es un dispositivo que permite aumentar la	No

	humedad de un ambiente.	
Luz de cultivo	Es una fuente de luz artificial.	No

Para ***Tabla VII: Identificar los componentes necesarios para el circuito*** se tomaron en cuenta los tipos de sensores contemplados en otras investigaciones [9] [10] [8], con respecto a los actuadores solo se tomaron en cuenta los ventiladores para permitir una mayor ventilación dentro de la Greenhouse y un servo motor que serviría para la apertura y cierre de la ventana, el motivo por el cual se excluyó a los humidificadores es debido a la alta humedad de la región, la cual se pudo apreciar con un medidor de humedad internamente en la greenhouse, con respecto a la exclusión de las luces de cultivo, esto fue debido a que se midió la cantidad de luz filtrada que llegaba a la greenhouse y esta resulto ser alta desde las 8 a.m. hasta las 3 p.m.

La ***Tarea I-4: Evaluar y seleccionar los componentes a utilizar en el circuito*** consistía en evaluar y seleccionar cuales son los componentes que se utilizarían para el circuito, tomando en cuenta los datasheets brindados por los proveedores y la disponibilidad del mercado para conseguir los mismo en Perú. Por los mismo se presentan las siguientes tablas:

Tabla X: Lista de sensores

Tipo	Componente	Precio
Temperatura y humedad ambiental	Sensor de temperatura y humedad SHT3x-DIS	S/40.00
	Sensor de temperatura y humedad relativa DHT22 (AM2302)	S/20.00
	Sensor de temperatura analógico LM35	S/5.00
	Sensor de temperatura digital DS18B20	S/11.00
	Sensor de temperatura y humedad relativa DHT21(AM2301)	S/25.00
	Sensor de temperatura RTD PT100 (2 HILOS)	S/15.00

Humedad de suelo	Sensor de humedad de suelo capacitivo	S/27.00
	Sensor de humedad de suelo FC-28	S/10.00
Lluvia	Sensor de lluvia y nieve FC-27	S/10.00
Luminosidad	Módulo sensor de luz digital BH1750	S/15.00

Para la evaluación de cada sensor se tomaron los dos factores ya mencionados anteriormente los cuales serían la disponibilidad de los componentes en el mercado peruano y los datasheets o fichas de especificaciones que las da cada proveedor de componentes. En este caso los sensores a elegir fueron los siguientes:

- **Sensor de temperatura y humedad SHT31x-DIS** [40]

  a. Especificaciones técnicas:

  - Voltaje de Operación: 2.4V a 5.5V DC
  - Interfaz de comunicación: I2C
  - Dirección I2C: 0x44
  - Rango de trabajo Temperatura: -40° a 125°C
  - Resolución Temperatura: 0.015°C
  - Precisión Temperatura: 0.2°C
  - Rango de trabajo Humedad: 0 a 100% RH
  - Resolución HR: 0.01 %RH
  - Precisión HR: 2% RH
  - Tiempos de muestreo rápidos

  b. Conexión:

  - VIN: +3.3V/+5V DC
  - GND: Tierra 0V
  - SCL: I2C Clock
  - SDA: I2C Data

- **Sensor de temperatura y humedad relativa DHT22 (AM2302)** [41]

  a. Especificaciones técnicas:

  - Voltaje de Operación: 3V - 6V DC
  - Rango de medición de temperatura: -40°C a 80 °C
  - Precisión de medición de temperatura: <±0.5 °C
  - Resolución Temperatura: 0.1°C
  - Rango de medición de humedad: De 0 a 100% RH
  - Precisión de medición de humedad: 2% RH
  - Resolución Humedad: 0.1%RH

- Tiempo de sensado: 2s
- Interface digital: Single-bus (bidireccional)
- Modelo: AM2302
- Dimensiones: 20*15*8 mm
- Peso: 3 gr.
- Carcasa de plástico blanco

b. Pines:

- Alimentación: +5V (VCC)
- Datos (DATA)
- No Usado (NC)
- Tierra (GND)

- **Sensor de temperatura analógico LM35** [42]

a. Especificaciones técnicas:

- Voltaje de Operación: 4V – 30V (5V recomendado)
- Rango de Trabajo: -55℃ hasta +150℃
- Precisión en el rango de -10°C hasta +85°C: ±0.5°C
- Pendiente: 10mV / ºC
- Bajo consumo energético: 60uA
- No necesita componentes adicionales
- Pines: +VCC, V salida, GND
- Baja impedancia de salida

- **Sensor de temperatura digital DS18B20** [43]

a. Especificaciones técnicas:

- Voltaje de operación: 3.0V – 5.5V DC
- Rango de medición: -55℃ hasta +125℃ (-67°F a +257°F)
- Precisión en el rango de -10°C hasta +85°C: ±0.5°C.
- Resolución ADC seleccionable de 9-12 bits
- Cables: Rojo (+VCC), Blanco (DATA 1-Wire), Negro (GND)
- Protocolo 1-Wire, solo necesita 1 pin para comunicarse
- Identificación única de 64 bits
- Cubierta de acero inoxidable de alta calidad, previene la oxidación de la sonda
- Sonda a prueba de agua
- Longitud de cable: 1m
- Dimensiones sonda: D5mm*L50mm
- Peso: 23 gramos

- **Sensor de temperatura y humedad relativa DHT21(AM2301)** [44]

  a. Especificaciones técnicas:

  - Voltaje de Operación: 3.5V - 5.5V DC
  - Consumo corriente: 1mA - 1.5mA
  - Rango de Temperatura: -40 hasta 80°C
  - Precisión Temperatura: +- 0.5°C
  - Resolución Temperatura: 0.1°C
  - Rango de Humedad Relativa: 0 a 100% RH
  - Precisión HR: +- 3%
  - Resolución Humedad: 0.1%RH
  - Tiempo de sensado: 2s
  - Interface digital: Single-bus (bidireccional)
  - Modelo: AM2301
  - Dimensiones: 60*28*13mm
  - Peso: 17 gr.
  - Carcasa de plástico negro
  - Longitud cable: 50cm

  b. Cables:

  - Rojo: Alimentación + (5 VDC)
  - Negro: Tierra (0 VDC)
  - Amarillo: Datos digitales I/O

- **Sensor de temperatura RTD PT100 (2 HILOS)** [45]

  a. Especificaciones técnicas:

  - Rango de trabajo: -100℃ hasta +400℃
  - Conexión: 3 Hilos
  - Longitud de cable: 1m
  - Dimensiones: D5mm x L100mm
  - Diámetro de la rosca: 8mm/0.31"
  - Material de sonda: acero inox.
  - Resistente al agua (la parte del sensor, no del cable)

- **Sensor de humedad de suelo FC-28** [46]

  a. Especificaciones técnicas:

  - Voltaje de alimentación: 3.3V - 5V DC (VCC)
  - Corriente de operación: 35mA
  - Voltaje de señal de salida analógico (AO) : 0 a VCC
  - Voltaje de señal de salida digital (DO) : 3.3V/5V TTL
  - Opamp LM393 en modo comparador, umbral (threshold) regulable por potenciómetro
  - Superficie de electrodo: Estaño
  - Incluye: Electrodo, Placa y cable de conexión
  - Vida útil electrodo sumergido: 3 a 6 meses

- Dimensiones YL-38: 30*16 mm
- Dimensiones YL-69: 60*20*5 mm

b. Conexiones:

- VCC: Voltaje de alimentación (3.3V - 5V DC)
- GND: Tierra (GND 0V)
- DO: Salida digital
- AO: Salida analógica

- **Sensor de humedad de suelo capacitivo** [46]

a. Especificaciones técnicas:

- Voltaje de alimentación: 3.3V - 5V DC
- Corriente operación: 5mA
- Voltaje de la señal de salida: 0 a 5V (Analógico)
- Modelo: capacitive soil moisture sensor v1.2
- Vida útil: 3 años mín.
- Conector: PH2.0-3P
- Incluye: Electrodo y cable jumper hembra
- Dimensiones: 98*23 mm
- Peso: 15 gramos

b. Conexiones:

- GND: Tierra (GND 0V)
- VCC: Voltaje de alimentación (3.3V - 5V DC)
- AOUT: Salida analógica

- **Sensor de lluvia y nieve FC-37** [46]

a. Especificaciones técnicas:

- Voltaje de Alimentación: 3.3V - 5V
- Voltaje de la señal de salida: 0~5V (Analógico)
- Salida digital de comparador: TTL
- Corriente: 15mA
- Tamaño: 50x40mm
- Superficie de electrodo: Estaño y nickel

- **Sensor fotorresistencia LDR 5528** [46]

a. Especificaciones técnicas:

- Sensor: LDR GL5528
- Resistencia en luz (10 lux): 8K-20K Ohm
- Resistencia en oscuridad: 1M Ohm
- Voltaje máx: 150V

- Potencia máx: 100mW
- Material fotosensible: CdS (Sulfato de Sodio)
- Frecuencia de luz pico: 540 nm

- **LDR 5528 módulo sensor de luz digital BH1750** [46]

  a. Especificaciones técnicas:

  - Voltaje de Operación: 3V – 5V
  - Interfaz digital a través de bus I2C con capacidad de seleccionar entre 2 direcciones
  - Respuesta espectral similar a la del ojo humano
  - Realiza mediciones de iluminancia y convierte el resultado a una palabra digital
  - Amplio rango de medición 1-65535 lux
  - Modo de bajo consumo de energía
  - Rechazo de ruido a 50/60 Hz
  - Baja dependencia de la medición contra la fuente de luz: halógeno, led, incandescente, luz de día, etc.

Con respecto a los módulos de comunicación y placa se cuenta con las siguientes especificaciones:

- **NODEMCU V2 ESP8266 WIFI** [47]

  a. Especificaciones técnicas:

  - Voltaje de Alimentación: 5V DC
  - Voltaje de Entradas/Salidas: 3.3V DC (No usar 5V)
  - Placa: NodeMCU v2 (Amica)
  - Chip conversor USB-serial: CP2102
  - SoM: ESP-12E (Ai-Thinker)
  - SoC: ESP8266 (Espressif)
  - CPU: Tensilica Xtensa LX3 (32 bit)
  - Frecuencia de Reloj: 80MHz/160MHz
  - Instruction RAM: 32KB
  - Data RAM: 96KB
  - Memoria Flash Externa: 4MB
  - Pines Digitales GPIO: 17 (4 pueden configurarse como PWM a 3.3V)
  - Pin Analógico ADC: 1 (0-1V)
  - Puerto Serial UART: 2
  - Certificación FCC
  - Antena en PCB
  - 802.11 b/g/n
  - Wi-Fi Direct (P2P), soft-AP
  - Stack de Protocolo TCP/IP integrado
  - PLLs, reguladores, DCXO y manejo de poder integrados

- Potencia de salida de +19.5dBm en modo 802.11b
- Corriente de fuga menor a 10uA
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Consumo de potencia Standby < 1.0mW (DTIM3)
- Pulsador RESET y FLASH
- Led indicadores: 2
- Dimensiones: 49*26*12 mm
- Peso: 9 gramos

b. Conectividad:

- SDIO 2.0, SPI, UART
- Integra RF switch, balun, 24dBm PA, DCXO y PMU
- Posee un procesador RISC, memoria en chip e interface para memoria externa
- Procesador MAC/Baseband integrado
- Interface I2S para apliaciones de audio de alta calidad
- Reguladores de voltaje lineales "low-dropout" en chip
- Arquitectura propietaria de generacion de clock "spurious free"
- Módulos WEP, TKIP, AES y WAPI integrados

- **NODEMCU-32 30-PIN ESP32 WIFI** [48]

a. Especificaciones técnicas:

- Voltaje de Alimentación: 5V DC
- Voltaje de Entradas/Salidas: 3.3V DC (No usar 5V)
- Placa: NodeMCU v2 (Amica)
- Chip conversor USB-serial: CP2102
- SoM: ESP-12E (Ai-Thinker)
- SoC: ESP8266 (Espressif)
- CPU: Tensilica Xtensa LX3 (32 bit)
- Frecuencia de Reloj: 80MHz/160MHz
- Instruction RAM: 32KB
- Data RAM: 96KB
- Memoria Flash Externa: 4MB
- Pines Digitales GPIO: 17 (4 pueden configurarse como PWM a 3.3V)
- Pin Analógico ADC: 1 (0-1V)
- Puerto Serial UART: 2
- Certificación FCC
- Antena en PCB
- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Stack de Protocolo TCP/IP integrado
- PLLs, reguladores, DCXO y manejo de poder integrados
- Potencia de salida de +19.5dBm en modo 802.11b
- Corriente de fuga menor a 10uA

- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Consumo de potencia Standby < 1.0mW (DTIM3)
- Pulsador RESET y FLASH
- Led indicadores: 2
- Dimensiones: 49*26*12 mm
- Peso: 9 gramos

b. Conectividad:

- SDIO 2.0, SPI, UART
- Integra RF switch, balun, 24dBm PA, DCXO y PMU
- Posee un procesador RISC, memoria en chip e interface para memoria externa
- Procesador MAC/Baseband integrado
- Interface I2S para apliaciones de audio de alta calidad
- Reguladores de voltaje lineales "low-dropout" en chip
- Arquitectura propietaria de generacion de clock "spurious free"
- Módulos WEP, TKIP, AES y WAPI integrados

- **Arduboard UNO R3** [18]

a. Especificaciones técnicas:

- Microcontrolador: ATmega328P (8-bit)
- Chip USB: ATmega16U2
- Conector USB: Tipo B
- Voltaje de operación: 5V DC
- Voltaje de alimentación: 6V - 20V DC(7-12V recomendado)
- Pines digitales I/O: 14 (6 salidas PWM)
- Entradas analógicas: 6 (ADC 10-bit)
- Corriente entrada/salida por pin: 40mA máx.
- Memoria FLASH: 32KB (2KB usados por el Bootloader)
- Memoria SRAM: 2KB
- Memoria EEPROM: 1KB
- Frecuencia de reloj: 16MHz
- Leds indicadores: ON, L(Pin 13), TX y RX
- Diseño compatible con Arduino® Uno R3
- Procedencia: China
- Incluye: Cable USB 30cm
- Dimensiones: 73*53*13 mm
- Peso: 30 gramos

- **Arduboard MEGA 2560 R3** [18]

a. Especificaciones técnicas:

- Microcontrolador: ATmega2560 (8-bit)
- Chip USB: ATmega16U2
- Conector USB: Tipo B

- Voltaje de operación: 5V DC
- Voltaje de alimentación: 6V - 20V DC(7-12V recomendado)
- Pines digitales I/O: 54 (15 salidas PWM)
- Entradas analógicas: 16 (ADC 10-bit)
- Corriente entrada/salida por pin: 40mA máx.
- Memoria FLASH: 256KB
- Memoria SRAM: 8KB
- Memoria EEPROM: 4KB
- Frecuencia de reloj: 16MHz
- Diseño compatible con Arduino® Mega 2560 R3
- Procedencia: China
- Incluye: Cable USB 30cm
- Dimensiones: 108*53*13 mm

Tomando en cuenta las especificaciones ya mencionadas, en la siguiente tabla se muestra la selección y evaluación de los sensores basados en criterios de evaluación adaptados a la investigación [49]:

Tabla XI: Selección y evaluación de sensores

Componente	**Criterios**					**Puntuación**
	1 pt: Baja 2 pt: Media 3 pt: Alta					
Sensores de Temperatura y Humedad	**Precisión**	**Rango**	**Resolución**	**Tiempo de sensado**	**Acople con otros dispositivos**	
Sensor de temperatura y humedad SHT31x-DIS	3	3	3	3	3	15
Sensor de temperatura y humedad relativa DHT22 (AM2302)	2	1	2	2	3	10
Sensor de temperatura analógico LM35	2	2	2	1	3	10
Sensor de temperatura digital DS18B20	2	2	2	1	3	10
Sensor de temperatura y humedad relativa DHT22 (AM2302)	2	2	2	1	3	10
Sensor de temperatura y humedad relativa DHT21(AM2301)	2	2	2	1	3	10
Sensor de temperatura RTD PT100 (2 HILOS)	3	3	3	2	3	14
Sensores de Humedad de Suelo	**Conectividad**		**Vida útil**	**Acople con otros dispositivos**		**Puntuación**
Sensor de humedad de suelo FC-28	3		3	3		9
Sensor de humedad de suelo capacitivo	3		2	3		8
Sensor de Lluvia	**Conectividad**		**Vida útil**	**Acople con otros dispositivos**		**Puntuación**
Sensor de lluvia y nieve FC-27	3		2	3		8
Sensores de Luz	**Rango**		**Conectividad**	**Acople con otros dispositivos**		**Puntuación**
LDR 5528 módulo sensor de luz digital BH1750	3		3	3		9
Sensor fotorresistencia LDR 5528	1		3	3		7
Módulos de Comunicación y Placas	**Conectividad**		**Vida útil**	**Acople con otros dispositivos**		**Puntuación**
NODEMCU V2 ESP8266 WIFI	2		3	3		8
NODEMCU-32 30-PIN ESP32 WIFI	3		3	3		9
Arduboard UNO R3	3		3	3		9
Arduboard MEGA 2560 R3	3		3	3		9

Como se muestra en la ***Tabla XI: Selección y evaluación de sensores*** se calificaron los sensores de temperatura y humedad, primeramente, con el mayor puntaje y mejores características se encontró al Sensor de temperatura y humedad SHT31x-DIS con una puntuación de 15, por lo cual se decidió seleccionar a este sensor ya posee la mayor precisión y rango en un menor tiempo de sensado.

Después se pasó a los sensores de humedad, de los cuales se eligió al sensor de humedad de suelo FC-28, primeramente, debido a su facilidad para conseguirlo en el mercado y otro punto importante es que sus diferencias con el otro sensor son mínimas.

Para el caso del sensor de lluvia, solo se encontró el Sensor de lluvia y nieve FC-27 en las tiendas que proveen al Perú, a pesar de ellos, este modelo cumple con su funcionalidad de manera eficaz.

Luego de entre los dos sensores de luz se eligió al LDR 5528 módulo sensor de luz digital BH1750, ya que conto con un rango por mucho mayor para la detección del nivel de luz.

Por último, los modelos de placas y módulos, el módulo para la comunicación WIFI seleccionado fue NODEMCU-32 30-PIN ESP32 WIFI, conocido mayormente como módulo ESP32, este a diferencia de su antecesor NODEMCU V2 ESP8266 WIFI, es de mayor calidad y más actualizado. Como placa se escogió a Arduboard UNO R3 debido a que cuenta con la cantidad de puertos óptima para poder trabajar con ella y las demás características necesarias, tales como el voltaje, pines, entradas analógicas y digitales.

```
// Libraries
#include <Wire.h>
#include <BH1750.h>
#include <Arduino.h>
#include "Adafruit_SHT31.h"
#include <Servo.h>
#include <AFMotor.h>
#include <ArduinoJson.h>

#define TIMEOUT 10000 // 10 sec

//--------------- ACTUATORS ---------------
// Servo motor
Servo motor;

// Fans
#define fan1 8  // Fan N°1
#define fan2 9  // Fan N°2
#define fan3 10 // Fan N°3
#define fan4 11 // Fan N°4
#define fan5 12 // Fan N°5

//--------------- SENSORS ---------------
// SHT31 - Temperature and humidify sensor
Adafruit_SHT31 sht31 = Adafruit_SHT31();

// BH1750 - Light sensor
BH1750 lightMeter;

JsonArray actions;
```

Evidencia I: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores–Parte *1*

La ***Evidencia I: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores -1*** corresponde al código para la obtención de los datos de los sensores mediante Arduino, así como las funcionalidades de los actuadores. En primer lugar, se encuentra la importación de la librería las cuales sirven para lo siguiente:

- **Wire.h:** Permite comunicarse a un dispositivo a Arduino mediante el protocolo I2C, el cual es un puerto y protocolo de comunicación serial, define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales, en este caso se utilizó para la comunicación de el sensor de temperatura y de luminosidad con la placa de Arduino.
- **BH1750.h:** Esta librería permitió la lectura de los datos obtenidos por el sensor BH1750, el cual es el sensor de luminosidad.
- **Arduino.h:** Se utiliza para la compatibilidad con otros dispositivos.
- **Adafruit_SHT31.h:** Esta librería permitió la lectura de los datos obtenidos por el sensor SHT31, el cual es el sensor de temperatura y humedad relativa.
- **Servo.h:** Sirve para el control del servomotor.
- **AFMotor.h:** Sirve para el control del servomotor.
- **ArduinoJson.h:** Esta librería permite la JSON serialización, JSON deserialización.

Luego se definen a los actuadores en este caso los ventiladores y el servo motor, posteriormente la declaración de los dos sensores que hacen uso del protocolo I2C,

es decir el sensor de temperatura y humedad relativa (SHT31) y el sensor de luminosidad (BH1750).

```
void setup ()
{
 Serial.begin(9600);

 // Fan - Ventilador
 fansSetUp();

 // Light sensor - Sensor de luz
 lightSensorSetUp();

 // Temperatura and humdify sensor - Sensor de temperatura y humedad
 thSensorSetUp();

 // Leer del EEPROM
 bool window_state = false;
 bool fan_state = false;
 bool led_state = false;
}
```

Evidencia II: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores–Parte *2*

La ***Evidencia II: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores - 2*** corresponde al código de la función setup() la cual realiza las configuraciones iniciales al momento de ejecutar el programa.

```
// Setup fans
void fansSetUp()
{
 pinMode(fan1, OUTPUT);
 pinMode(fan2, OUTPUT);
 pinMode(fan3, OUTPUT);
 pinMode(fan4, OUTPUT);
 pinMode(fan5, OUTPUT);
}

// Setup light sensor
void lightSensorSetUp()
{
 Wire.begin();
 lightMeter.begin();
}

// Setup temperature sensor
void thSensorSetUp()
{
 if (!sht31.begin(0x44))
   Serial.println("No se encontro el sensor SHT31");
}
```

Evidencia III: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores–Parte *3*

En la ***Evidencia III: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores – 3*** se muestran las funciones que son llamadas dentro de la función **setup().**

```
// void loop()
{
 // Light sensor - Sensor de luz
 float light = readLight();

 // Temperatura and humidify sensor - Sensor de temperatura y humedad
 float temperature = readTemperature();
 float humidify = readHumidify();

 // Earth humidify - Humedad de suelo
 int soilMoisture = readSoilMoisture();

 // Rain sensor - Sensor de lluvia
 int rain = readRain();

 // Send to ESP32
 String jsonData = "{";
 jsonData += "\"light\": " + String(light, 2);
 jsonData += ", \"temperature\": " + String(temperature, 2);
 jsonData += ", \"humidify\": " + String(humidify, 2);
 jsonData += ", \"soilMoisture\": " + String(soilMoisture);
 jsonData += ", \"rain\": " + String(rain);
 jsonData += "}";
 Serial.println(jsonData);

 delay(TIMEOUT);

 // Recibir la acción
 // actions = accion_recibida;
 StaticJsonDocument<JSON_ARRAY_SIZE(10)> doc;
 if (Serial.available())
 {
  String response = Serial.readString();
  int indexError = response.indexOf("Error: ");
  if (indexError != -1)
  {
   Serial.println(response.substring(indexError, response.indexOf(".") + 1));
   return;
  }

  deserializeJson(doc, response);
  actions = doc.as<JsonArray>();

  Serial.print("Acciones detectadas:");
  serializeJson(doc, Serial);

  // Window - Ventana
  window();

  // Fan - Ventilador
  fans();
 }
}
```

Evidencia IV: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores–Parte *4*

En la ***Evidencia IV: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores – 4*** se muestra función loop() en la cual se ejecutan cada 10 segundos según el TIMEOUT establecido, las funciones para

obtención de los datos de los sensores así como la forma en la cual se traerán las acciones a realizar por los actuadores desde el módulo de WIFI ESP32.

```
// Read light sensor
float readLight()
{
 float light = lightMeter.readLightLevel();
 if (light < 0)
   return -1;
 return light;
}

// Read temperature sensor
float readTemperature()
{
 float temperature = sht31.readTemperature();
 if (isnan(temperature))
   return -1;
 return temperature;
}

// Read humidity sensor
float readHumidify()
{
 float humidify = sht31.readHumidity();
 if (isnan(humidify))
   return -1;
 return humidify;
}

// Read rain sensor
int readRain()
{
 int analogValueRain = analogRead(A3); // Rain values range -> 0 - 1023
 return analogValueRain;
}

float readSoilMoisture()
{
 int sm1 = analogRead(A0); // Soil Moisture 1
 int sm2 = analogRead(A1); // Soil Moisture 2
 int sm3 = analogRead(A2); // Soil Moisture 3

 return map((sm1 + sm2 + sm3) / 3, 1023, 0, 0, 100); // Porcentaje
}
```

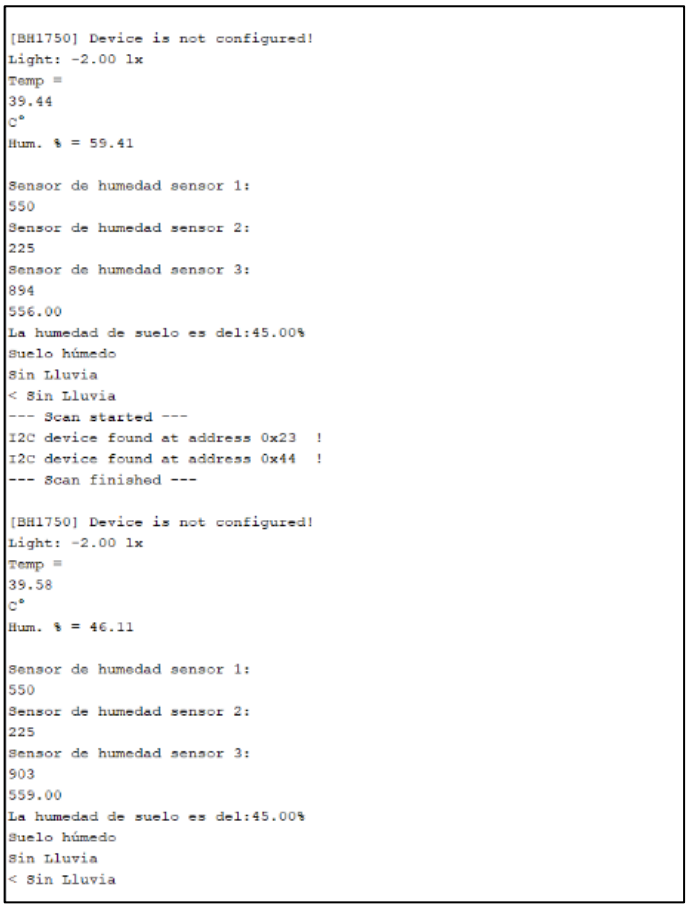
Evidencia V: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores–Parte 5

En la ***Evidencia V: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores - 5 se*** muestran las funciones que se utilizaron para la obtención de los datos de los sensores, las cuales son llamadas en la función ***loop().***

```
void fans()
{
 bool ON_FAN = findAction("ON_FAN");
 bool OFF_FAN = findAction("OFF_FAN");

 bool fanAction;
 if (ON_FAN)
 {
  fanAction = true;
  Serial.println("Ventiladores encendidos");
 }

 if (OFF_FAN)
 {
  fanAction = false;
  Serial.println("Ventiladores encendidos");
 }

 digitalWrite(fan1, fanAction);
 digitalWrite(fan2, fanAction);
 digitalWrite(fan3, fanAction);
 digitalWrite(fan4, fanAction);
 digitalWrite(fan5, fanAction);
}

void window()
{
 bool OPEN_WINDOW = findAction("OPEN_WINDOW");
 bool CLOSE_WINDOW = findAction("CLOSE_WINDOW");
 Serial.println(OPEN_WINDOW);
 Serial.println(CLOSE_WINDOW);

 if (OPEN_WINDOW)
   openWindow(10);
 if (CLOSE_WINDOW)
   closeWindow(10);
}

void openWindow(byte velocity)
{
 motor.attach(6);
 for (byte i = 0; i < 140; i = i + velocity)
 {
  motor.write(i);
  delay(500);
 }
 motor.detach();
 Serial.println("Ventana abierta");
}

void closeWindow(byte velocity)
{
 motor.attach(6);
 for (byte i = 140; i >= 0; i = i - velocity)
 {
  motor.write(i);
  delay(500);
 }
 motor.detach();
 Serial.println("Ventana cerrada");
}

bool findAction(String action)
{
 for (JsonVariant v : actions)
 {
  String a = v.as<String>();
  if (a == action){
    Serial.println(a);
    return true;
    }

 }
 return false;
}
```

Evidencia VI: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores –Parte 6

En la ***Evidencia VI: Desarrollo de la obtención de los datos ambientales y funcionalidad de los actuadores – 6*** se encuentran la función ***findAction()***, que se utiliza para conocer qué acciones envía el servidor a Arduino, dependiendo de la acción se las funciones ***Window()*** y ***Fans()***, en el caso de ***Window()*** ejecuta la función ***openWindow()*** para abrir y ***closeWindow()*** para cerrar la ventana.



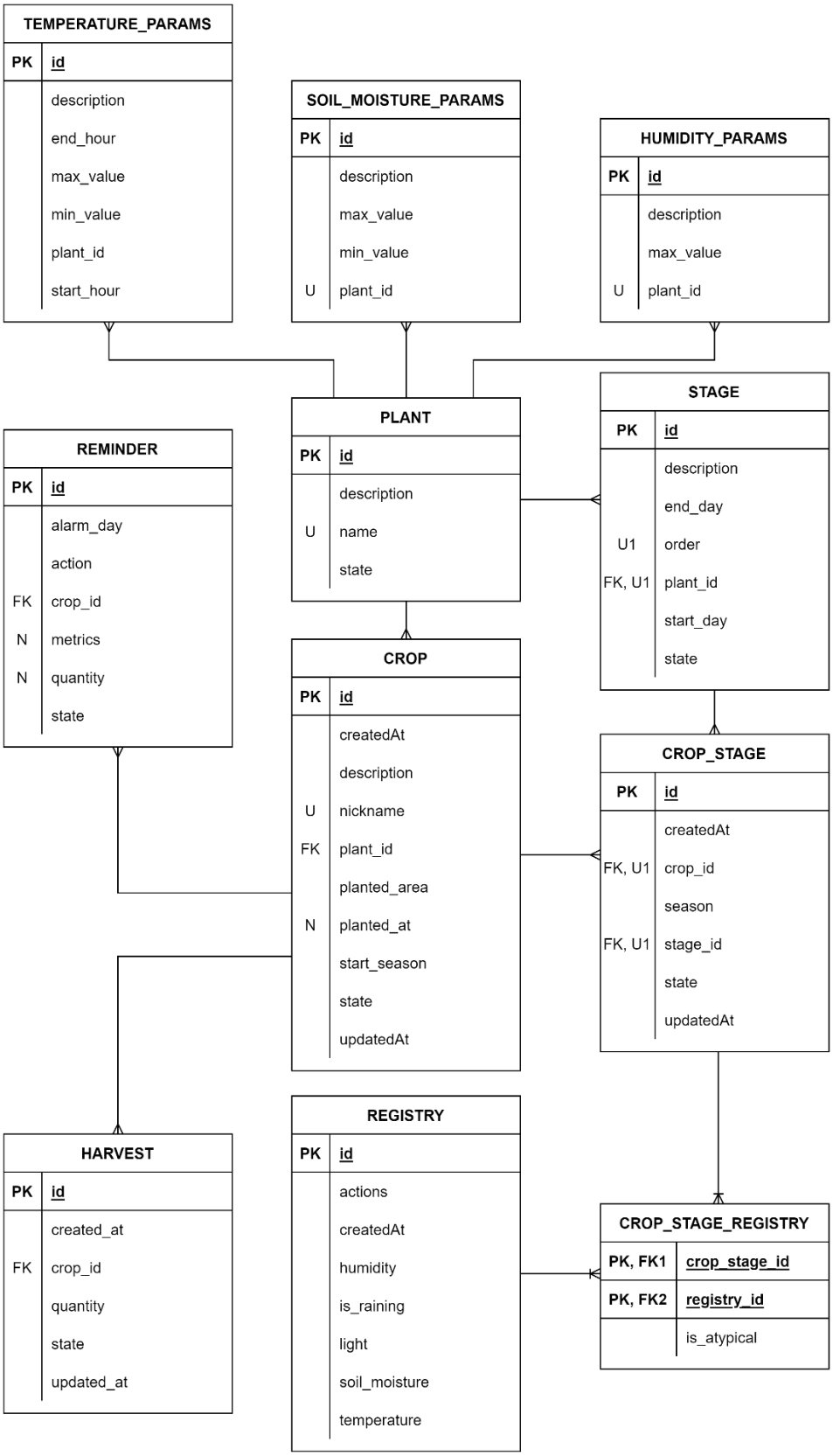
Evidencia VII: Captura de datos de los sensores

En la ***Evidencia I - VII: Captura de datos de los sensores*** se muestra como mediante el monitor serial del IDE de Arduino se capturaron los datos de los sensores en tiempo real.

El circuito diseñado se evidencia en el ***Anexo N°4*** y desarrollado e implementado en el ***Anexo N°6***

4.1.2. Sprint 2: Desarrollar la Base de Datos

A continuación, en la Figura se muestra el Modelo Entidad Relación de la Base de datos desarrollada:



Evidencia VIII: Modelo entidad relación

```
//Connection
import Knex from 'knex';
import { knexLittleLogger } from 'knex-little-logger';
import config from '../../knexfile';

const connection = knexLittleLogger(Knex(config[process.env.NODE_ENV]), {
        bindings: true,
});

export default connection;
```

Evidencia IX: Configuración del Servidor de Base de datos

En la ***Evidencia IX: Configuración del Servidor de Base de datos*** se el archivo de conexión, sin embargo, para el desarrollo del servidor se trabajó con los siguientes archivos:

a. **Models:** Describe el conjunto de datos posibles. En este caso se crearon:
   - Crop.ts
   - CropStage.ts
   - CropStageRegistry.ts
   - Greenhouse.ts
   - Harvest.ts
   - HumidityParamas.ts
   - Plant.ts
   - Registry.ts
   - Reminder.ts
   - SoilMoistureParams.ts
   - Stage.ts
   - TemperatureParams.ts

b. **Resolvers:** Los cuales son funciones que se encargan de completar los datos de un solo campo en su esquema. En este caso se crearon:
   - Crop.resolver.ts
   - CropStage.resolver.ts
   - CropStageRegistry.resolver.ts
   - Greenhouse.resolver.ts
   - Harvest.resolver.ts
   - HumidityParamas.resolver.ts
   - Plant.resolver.ts
   - Registry.resolver.ts
   - Reminder.resolver.ts
   - SoilMoistureParams.resolver.ts
   - Stage.resolver.ts
   - TemperatureParams.resolver.ts

c. **Inputs:** Es una convención para el tipo de entrada

   - Crop.input.ts
   - CropStage.input.ts

- CropStageRegistry.input.ts
- Greenhouse.input.ts
- Harvest.input.ts
- HumidityParamas.input.ts
- Plant.input.ts
- Registry.input.ts
- Reminder.input.ts
- SoilMoistureParams.input.ts
- Stage.input.ts
- TemperatureParams.input.ts

```
/*****************************************************************************************************************
***********
 ESP32 like HTTP Client

*****************************************************************************************************************
***********/

/*
 Error codes:
 - Wifi:
    1: No conectado a Wifi.
 - Sensores:
    20: Los datos no son recibidos desde Arduino.
    21: No se recibió un JSON por parte del Arduino.
    41: Error al leer la humedad.
    42: Error al leer la lluvia.
    43: Error al leer la luz.
    44: Error al leer la humedad de suelo.
    45: Error al leer la temperatura.
 - HTTP:
    60: Error al hacer la consulta HTTP.
    61: Error de servidor HTTP.
    62: Sin respuesta.
*/
//#include "HardwareSerial.h"
#include <ESP32Ping.h>
#include <WiFi.h>
#include <HTTPClient.h> // nos permite hacer peticiones http
#include <ArduinoJson.h>
#define RX 16
#define TX 17
```

Evidencia X: Configuración del ESP32 WebSocket Client – Parte 1

En la ***Evidencia X: Configuración del ESP32 WebSocket Client – Parte 1*** se definen los errores de código, en caso ocurra algún error al momento de la ejecución o recepción de los datos, así mismo se importan las librerías las cuales son:

- **ESP32Ping.h:** Para poder comprobar el estado de la comunicación.
- **WiFi.h:** Para la conexión con el protocolo de WIFI
- **HTTPClient.h:** Permite realizar peticiones http
- **ArduinoJson.h:** Para trabajar con tipos de datos JSON

```
//HardwareSerial Serial2(2);
int i;

// credenciales de la red a la cual nos conectaremos
//const char* ssid = "Poli";
//const char* password = "12345699";

const char *ssid = "MANUEL CASTRO 2.4GHz_EXT";
const char *password = "1808M68C";

DynamicJsonDocument requestJSON(1024);

// Url's para hacer las peticiones
const char *graphqlEndpoint = "http://192.168.0.14:4000/graphql";
```

Evidencia XI: Configuración del ESP32 WebSocket Client – Parte 2

En la ***Evidencia XI: Configuración del ESP32 WebSocket Client – Parte 2*** se asignan las credenciales para poder conectarse con el Router, así como la declaración de un ***DinamicJsonDocument*** y las ***Url*** del api de Graphql para poder conectarnos y realizar las suscripciones, en las cuales se obtiene los datos de las variables ambientales cada minuto.

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

 //Comunicate with arduino one
 Serial2.begin(9600, SERIAL_8N1, RX, TX);

 // nos conectamos a la red
 connectToWifi();

 requestJSON["query"] = "mutation CreateRegistryMutation($data: RegistryCreateInput!) { create_registry(data: $data) { actions } }";
 requestJSON["operationName"] = "CreateRegistryMutation";
} // EOF setup
```

Evidencias XII: Configuración del ESP32 WebSocket Client – Parte 3

En la **Evidencias XII: Configuración del ESP32 WebSocket Client – Parte 3** se muestra la función ***setup()*** en la cual se realizan las configuraciones iniciales y se encuentra la comunicación serial entre la placa de Arduino uno y el módulo de WIFI ESP32, luego de ello se realiza la conexión con el WIFI, también se declara la ***query*** para realizar el registro de la mutación y el nombre de la operación.

```
void loop()
{
  Serial.println("==========================");

  String variables;
  if (Serial2.available() > 0)
    variables = Serial2.readString();

  // Serial.println(WiFi.localIP());

  if (WiFi.status() != WL_CONNECTED)
  {
    printError(1);
    return;
  }

  if (variables.length() <= 10)
  {
    printError(20);
    return;
  }

  variables = extractJson(variables);

  if (variables == "{}") return;

  Serial.print("Variables de Arduino: ");
  Serial.println(variables);

  DynamicJsonDocument sensorsData(1024);
  deserializeJson(sensorsData, variables);
  float humidify = validateData(sensorsData["humidify"]);
  float rain = validateData(sensorsData["rain"]);
  float light = validateData(sensorsData["light"]);
  float soilMoisture = validateData(sensorsData["soilMoisture"]);
  float temperature = validateData(sensorsData["temperature"]);

  if (humidify == -1)
  {
    printError(41);
    return;
  }
  if (rain == -1)
  {
    printError(42);
    return;
  }
  if (light == -1)
  {
    printError(43);
    return;
  }
  if (soilMoisture == -1)
  {
    printError(44);
    return;
  }
  if (temperature == -1)
  {
    printError(45);
    return;
  }

  // Lógica de obtención de datos de los sensores.
  requestJSON["variables"]["data"]["humidity"] = humidify;
  requestJSON["variables"]["data"]["rain"] = rain;
  requestJSON["variables"]["data"]["light"] = light;
  requestJSON["variables"]["data"]["soil_moisture"] = soilMoisture;
  requestJSON["variables"]["data"]["temperature"] = temperature;

  String bodyRequest;
  serializeJson(requestJSON, bodyRequest);
  String response = executeQuery(graphqlEndpoint, bodyRequest);
  Serial.print("Respuesta: ");
  Serial.println(response);

  if (response == "{}")
  {
    Serial.println("No se obtuvo respuesta del servidor");
    printError(62);
    return;
  }

  DynamicJsonDocument responseJSON(1024);
  deserializeJson(responseJSON, response);

  // Mandar acciones al arduino
  serializeJson(responseJSON["data"]["create_registry"]["actions"], Serial2);

  delay(REQUEST_INTERVAL);
}

void connectToWifi()
{
  WiFi.begin(ssid, password);
  Serial.println("Connecting");

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.print("Conectado a la red con la IP: ");
  Serial.println(WiFi.localIP());

  Serial.print("RRSI: ");
  Serial.println(WiFi.RSSI());
}
```

Evidencia XIII: Configuración del ESP32 WebSocket Client – Parte 4

En la ***Evidencia XIII: Configuración del ESP32 WebSocket Client – Parte 4*** se muestra la función ***loop()*** en la cual se obtienen los datos de las variables ambientales mediante la placa arduino y luego de ello se agregan a la ***query*** para posteriormente ejecutar esta y generar el registro de la suscripción. Teniendo la función ***loop()*** un TIMEOUT de 1 minuto, es decir que se repetira dejando ese intervalo de tiempo, por lo cual se registraran los datos tomando en cuenta ese tiempo.

```
String executeQuery(const char *url, const String body)
{
 HTTPClient http;
 http.begin(url);
 http.setTimeout(15000);
 http.addHeader("Content-Type", "application/json");

 // Enviamos peticion HTTP
 int httpResponseCode = http.POST(body);

 String res = "{}";
 Serial.print("Consulta: ");
 Serial.println(body);

 if (httpResponseCode > 0)
 {
  if (httpResponseCode == HTTP_CODE_OK)
  {
   res = http.getString();
  }
  else
  {
   Serial.print("Error code from server: ");
   Serial.println(httpResponseCode);
   printError(61);
  }
 }
 else
 {
  Serial.print("Error code: ");
  Serial.println(httpResponseCode);
  printError(60);
 }

 // liberamos
 http.end();

 return res;
}
```

Evidencia XIV: Configuración del ESP32 WebSocket Client – Parte 5

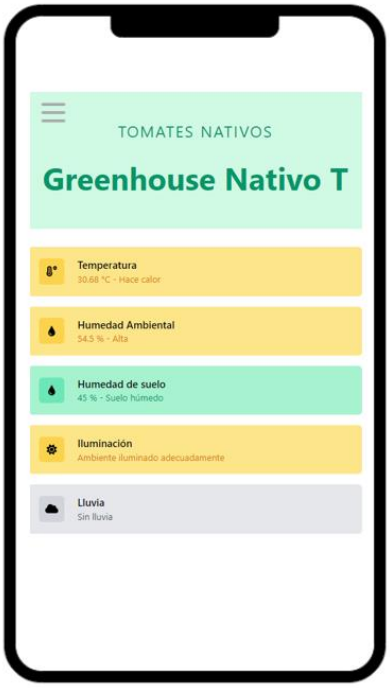
En ***la Evidencia XIV: Configuración del ESP32 WebSocket Client – Parte 5*** se encuentra la función ***executeQuery()*** se ejecuta la suscripción enviando una petición http, tal y como se puede apreciar en el código.

```
String extractJson(String data)
{
 int maxIndex = data.length() - 1;
 int a, b = -1;

 for (int i = 0; i <= maxIndex; i++)
 {
  if (data.charAt(i) == '{')
   a = i;
  if (data.charAt(i) == '}')
   b = i + 1;
 }

 if (a == -1 || b == -1)
 {
  printError(21);
  return "{}";
 }

 return data.substring(a, b);
}

float validateData(JsonVariant value)
{
 return value != nullptr ? value.as<float>() : -1;
}

void printError(int errorCode)
{
 Serial.print("Error: ");
 Serial.print(errorCode);
 Serial.println(".");

 Serial2.print("Error: ");
 Serial2.print(errorCode);
 Serial2.println(".");
}
```

Evidencia XV: Configuración del ESP32 WebSocket Client – Parte 6

Por último como parte de la programación perteneciente al módulo de WIFI ESP32, en la ***Evidencia XV: Configuración del ESP32 WebSocket Client – Parte 6*** se encuentran las funciones de ***extractJson()***, ***validateData()*** y ***printError()*** los cuales sirven respectivamente para lo que su nombre menciona, el primero de estos es para extraer el dato de tipo ***Json***, así como ver si este se encuentra vacío, la segunda para validar la data y la última para imprimir los errores que pueden ocurrir al momento de ejecutar.

4.1.3. Sprint 3: Gestionar el control y monitoreo del ambiente



Evidencia XVI: Desarrollo de las funcionalidades de la interfaz home - Interfaz

```
const SUSCRIPTION = gql`
  subscription registry_created {
    registry_created {
      id
      actions
      humidity
      light
      rain
      soil_moisture
      temperature
    }
  }
`;


const LATEST_REGISTRY = gql`
  query LatestRegistry {
    latest_registry {
      humidity
      light
      rain
      soil_moisture
      temperature
      actions
    }
  }
```

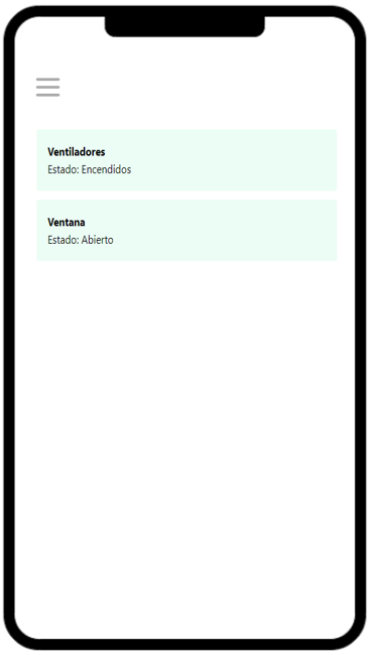
Evidencia XVII: Desarrollo de las funcionalidades de la interfaz home – Parte 1

Para poder mostrar los datos obtenidos por el servidor desde el módulo de WIFI, se realiza una suscripción y una ***query***, las cuales son las presentadas en el cuadro superior, la suscripción se realiza en caso se estén escuchando datos en tiempo real y se realiza la ***query*** de ***lastes_registry*** cuando no se están escuchando datos, entonces esta muestra el ultimo capturado, tal y como se muestra en ***Evidencia XVII: Desarrollo de las funcionalidades de la interfaz home - Interfaz.***

```
return (
  <div className="flex-grow bg-white">
    <Header />
    <br />
    <TemperatureItem
     temperature={dataLR.temperature}
     actions={dataLR.actions}
    />
    <HumidityItem humidity={dataLR.humidity} actions={dataLR.actions} />
    <SoilMoistureItem
     soilmoisture={dataLR.soil_moisture}
     actions={dataLR.actions}
    />
    <LightItem light={dataLR.light} actions={dataLR.actions} />
    <RainItem rain={dataLR.rain} actions={dataLR.actions} />
  </div>
 );
```

Evidencia XVIII: Desarrollo de las funcionalidades de la interfaz home – Parte 2

En la ***Evidencia XVIII: Desarrollo de las funcionalidades de la interfaz home – Parte 2***, se muestra lo que se debe retornar tras obtener la suscripción o el ultimo registro en caso haya habido alguna interrupción de la señal, se coloca la data dentro de una variable y luego de ello se llaman a los ítems correspondientes a cada variable para darles formato, en caso alguno de estos se encuentre fuera del rango estipulado en los parámetros, aparecerá esta advertencia dentro de los ítems.



Evidencia XIX: Desarrollo de las funcionalidades de la interfaz actuadores - Interfaz

En la ***Evidencia XIX: Desarrollo de las funcionalidades de la interfaz actuadores - Interfaz*** muestra el estado de los actuadores en tiempo real, tomando

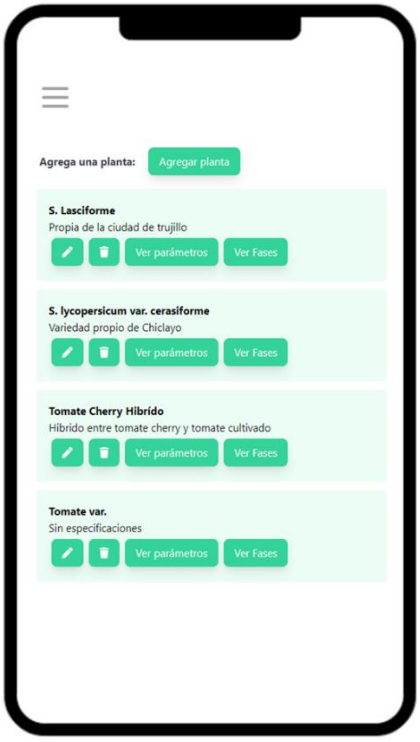
en cuenta la lógica estipulada en el servidor con los parámetros ingresados por el usuario para las plantas.

```
// Verificamos si los valores están dentro de los parámetros.
const isCold = data.temperature < minTemperatureParams;
const isHot = maxTemperatureParams < data.temperature;
const temperatureOK = !isCold && !isHot;
const humidityOK = data.humidity <= maxHumidity;
const dirtIsOverDry = data.soil_moisture < minSoilMoisture;
const dirtIsOverWet = maxSoilMoisture < data.soil_moisture;
const soilMoistureOK = !dirtIsOverDry && !dirtIsOverWet;
const isRaining = data.rain < 500; // Si el valor es por debajo de 500, entonces está lloviendo.


const closeWindow = () => {
actions.push(Action.CLOSE_WINDOW);
greenhouseState.window_opened = false;
greenhouseState.window_disabled_until_at = now
 .add(greenhouseState.window_cooldown, 'minutes')
 .toDate();
};

const openWindow = () => {
actions.push(Action.OPEN_WINDOW);
greenhouseState.window_opened = true;
greenhouseState.window_disabled_until_at = now
 .add(greenhouseState.window_cooldown, 'minutes')
 .toDate();
};

const turnOnFans = () => {
actions.push(Action.FANS_ON);
greenhouseState.fans_on = true;
greenhouseState.fans_disabled_until_at = moment()
 .add(greenhouseState.fans_cooldown, 'minutes')
 .toDate();
};

const turnOffFans = () => {
actions.push(Action.FANS_OFF);
greenhouseState.fans_on = false;
greenhouseState.fans_disabled_until_at = moment()
 .add(greenhouseState.fans_cooldown, 'minutes')
 .toDate();
};
```

Evidencia XX: Desarrollo de las funcionalidades de la interfaz actuadores

En la ***Evidencia XX: Desarrollo de las funcionalidades de la interfaz actuadores*** se muestra que para el funcionamiento de los actuadores, la lógica se llevó del lado del servidor, evaluando si los valores de las variables ambientales se encuentran o no dentro de los parámetros establecidos previamente, luego de ello se evalúa si es necesario o no cambiar el estado de estos, ya que en caso de que la temperatura se encuentre muy alta, y la humedad de suelo sea muy alta, la ventana deberá abrirse y los ventiladores encenderse, caso contrario o en caso de lluvia estos deben cerrarse y apagarse correspondientemente.

4.1.4. Sprint 4: Gestionar Registros



Evidencia XXI: Gestionar planta – Interfaz listar Plantas

Evidencia XXII: Gestionar planta – Interfaz de agregar y modificar planta

```
function Plant() {
 const { loading, error, data } = useQuery(PLANTS_QUERY);

 if (loading) return <p>Cargando ...</p>;
 if (error) return <p>Lo sentimos, pero hay errores: {error.message}</p>;

 return (
  <div className="flex-grow bg-white">
   <br />
   <div className="mb-4">
    <span className="py-4 px-4 text-gray-700 font-bold">
     Agrega una planta:
    </span>
    <Link
     className="py-2 px-4 m-1 text-white rounded-lg bg-green-400 shadow-lg block md:inline-block"
     to="/plants/create"
    >
     Agregar planta
    </Link>
   </div>
   {data.plants.map((plant) => (
    <PlantItem plant={plant} key={plant.id} />
   ))}
  </div>
 );
}
```

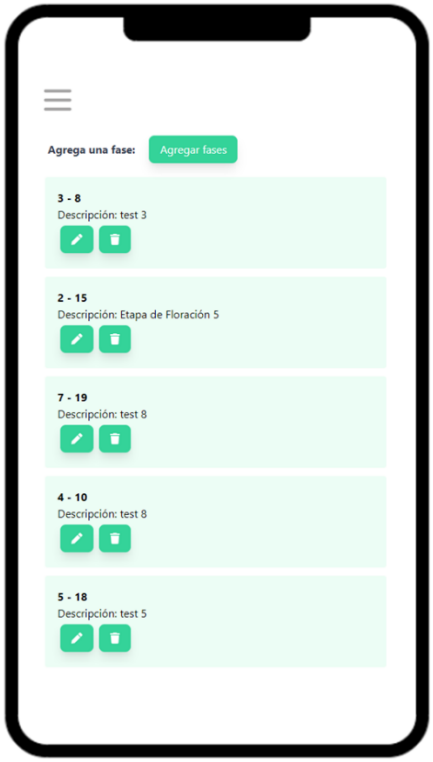
Evidencia XXIII: Gestionar planta – Listar Plantas

La ***Evidencia XXIII: Gestionar planta – Parte 1 Listar Plantas*** muestra que se realiza la ***query*** de para listar todas las plantas registradas, luego de ello, mapeamos la data resultante y lo pasamos al componente ***PlantItem***, el cual corresponde a cada ***Item*** de la lista de plantas que aparece en el aplicativo.

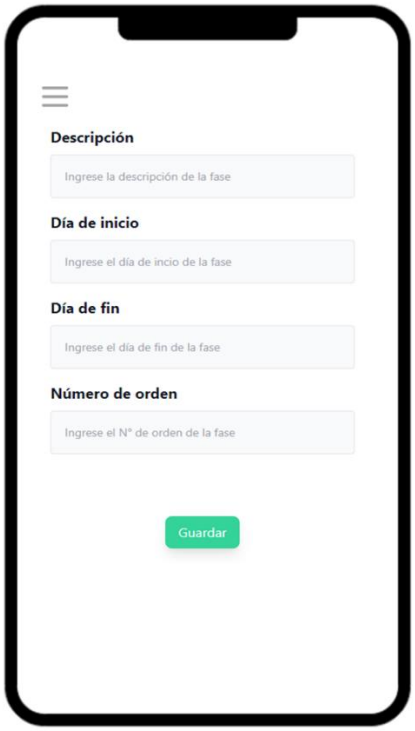
```
const { data, loading } = useQuery(GET_PLANT_QUERY, {
  variables: { id: Number(id) },
  skip: !id,
  update(cache, { data: { create_plant } }) {
    cache.modify({
      fields: {
        plants(existingPlants = []) {
          const newPlantRef = cache.writeFragment({
            data: create_plant,
            fragment: gql`
              fragment NewPlant on Plant {
                id
                type
              }
            `,
          });
          return [...existingPlants, newPlantRef];
        },
      },
    });
  },
});


const executeMutation = async (data) => {
  const variables = { data };
  if (id) {
    variables.id = Number(id);
    delete variables.data.stages;
  }
  try {
    const response = await mutation({
      variables,
    });
    if (response.data) navigate("/plants", { replace: true });
  } catch (error) {}
};
```

Evidencia XXIV: Gestionar planta –Registro y Modificación

En la ***Evidencia XXIV: Gestionar planta – Parte 1 Registro y Modificación*** se puede apreciar para poder realizar el registro o actualización de una planta es necesario verificar si es que se está pasando un parámetro, en caso sea así se realiza una actualización y se colocan los valores dentro de los inputs, caso contrario sería un registro de una planta, en este caso se ha contemplado la posibilidad de que una planta sea registrada con sus fases correspondientes.



Evidencia XXV: Gestionar Fases – Interfaz listar fases de una planta



Evidencia XXVI: Gestionar Fases – Interfaz Registro y modificación de la fase de una planta

```
export default function Stage() {
 const { id } = useParams();
 const { data, loading } = useQuery(STAGES_QUERY, {
  variables: { id: Number(id) },
 });

 const plant_id = id;

 if (loading) return <p>Cargando datos</p>;
 if (!data.plant?.stages) {
  <div className="flex-grow">
   <div className="mb-4">
    <span className="text-gray-700 font-bold">Agrega una fase:</span>
    <Link
     className="p-2 pl-5 pr-5 bg-transparent border-2 border-green-500 text-green-500 text-lg rounded-lg transition-colors duration-700
transform hover:bg-green-500 hover:text-gray-100 focus:border-4 focus:border-green-300"
     to={`/stage/create/${plant_id}`}
    >
     Agregar fases
    </Link>
   </div>
  </div>;
 }

 return (
  <div className="flex-grow bg-white">
   <br />
   <div className="mb-4">
    <span className="py-4 px-4 text-gray-700 font-bold">
     Agrega una fase:
    </span>
    <Link
     className="py-2 px-4 m-1 text-white rounded-lg bg-green-400 shadow-lg block md:inline-block"
     to={`/stage/create/${plant_id}`}
    >
     Agregar fases
    </Link>
   </div>
   {data.plant.stages.map((stage) => (
    <StageItem stage={stage} key={stage.id} />
   ))}
  </div>
 );
}
```

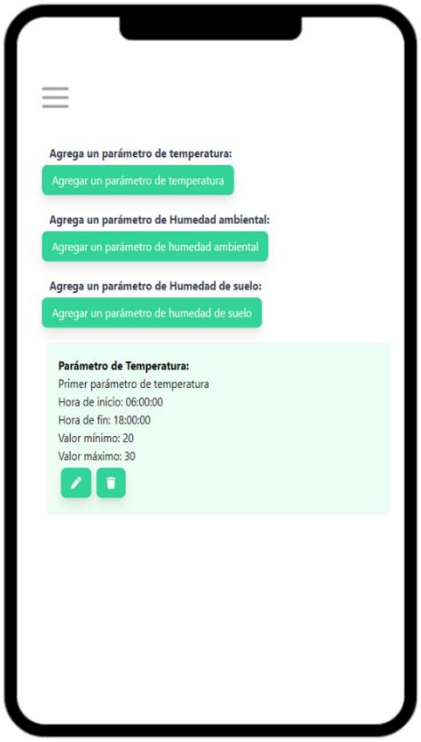
Evidencia XXVII: Gestionar Fases – Listar fases de una planta

En la ***Evidencia XXVIII: Gestionar Fases – Listar fases de una planta*** se muestra que las fases se listarán conforme a cada planta, por lo cual primero se realiza la ***query*** y luego se llama al componente, en este caso ***StageItem***, el cual corresponde a cada ***item*** de la lista que se aprecia.

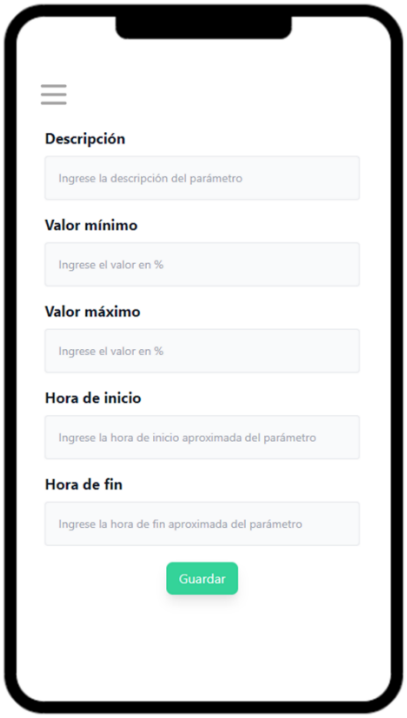
```
const { id } = useParams();
  const { register, handleSubmit, setValue } = useForm();
  const navigate = useNavigate();
  const location = useLocation();
  const createForm = location.pathname.includes("create");

  useQuery(GET_STAGE, {
    variables: { id: Number(id) },
    skip: createForm,
    onCompleted: (data) => {
      setValue("description", data.stage.description);
      setValue("start_day", data.stage.start_day);
      setValue("end_day", data.stage.end_day);
      setValue("order_number", data.stage.order_number);
    },
    update(cache, { data: { create_stage } }) {
      cache.modify({
        fields: {
          plants(existingstages = []) {
            const newstageRef = cache.writeFragment({
              data: create_stage,
              fragment: gql`
                fragment Newstage on stage {
                  id
                  type
                }
              `,
            });
            return [...existingstages, newstageRef];
          },
        },
      });
    },
  });

  const [mutation] = useMutation(
    createForm ? CREATE_STAGE_MUTATION : UPDATE_STAGE_MUTATION
  );

  const executeMutation = async (data) => {
    const variables = {
      data: {
        ...data,
        start_day: Number(data.start_day),
        end_day: Number(data.end_day),
        order_number: Number(data.order_number),
        plant_id: Number(id),
      },
    };
    if (!createForm) {
      variables.id = Number(id);
      delete variables.data.order_number;
      delete variables.data.plant_id;
    }

    try {
      const response = await mutation({
        variables,
      });
      if (response.data) {
        navigate(
          `/stage/${
            response.data.create_stage?.plant_id ||
            response.data.update_stage?.plant_id
          }`,
          {
            replace: true,
          }
        );
      }
    } catch (error) {
      console.error(error);
    }
  };
```

Evidencia XXVIII: Gestionar Fases –Registro y modificación de la fase de una planta

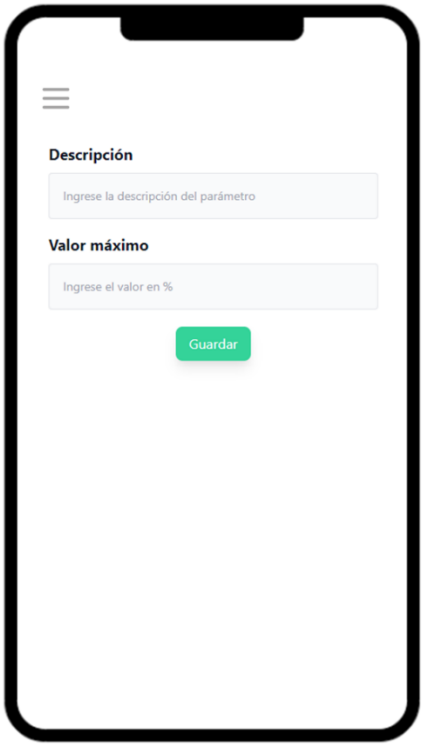
En la ***Evidencia XXVIII: Gestionar Fases –Registro y modificación de la fase de una planta*** al igual que para los otros formularios presentados, el registro o actualización de la fase de una planta requiere verificar si es que se está pasando un parámetro, en este caso el id del parámetro y que la referencia contenga la palabra "***create***", en caso sea así se realiza una el registro, caso contrario se realiza la actualización, colocando previamente los valores del parámetro dentro de los inputs.



Evidencia XXIX: Gestionar Parámetros - Interfaz de listar parámetros de una planta



Evidencia XXX: Gestionar Parámetros - Interfaz de registro y modificación del parámetro de temperatura de una planta



Evidencia XXXI: Gestionar Parámetros - Interfaz de registro y modificación del parámetro de humedad ambiental de una planta

Evidencia XXXII: Gestionar Parámetros - Interfaz registro y modificación del parámetro de humedad de suelo de una planta

```
export default function Parameter() {
  const { id } = useParams();
  const { data, loading } = useQuery(PLANT_QUERY, {
    variables: { id: Number(id) },
  });
  if (loading) return <p>Cargando ...</p>;
  console.log(data.plant);
  return (
    <div className="flex-grow bg-white">
      <br />
      <div className="mb-4">
        <span className="py-4 px-4 text-gray-700 font-bold">
          Agrega un parámetro de temperatura:
        </span>
        <Link
          className="py-2 px-4 m-1 text-white rounded-lg bg-green-400 shadow-lg block md:inline-block"
          to={`/temperatureparameter/create/${id}`}
        >
          Agregar un parámetro de temperatura
        </Link>
      </div>
      {!data.plant?.humidity_parameter && (
        <>
          <div className="mb-4">
            <span className="py-4 px-4 text-gray-700 font-bold">
              Agrega un parámetro de Humedad ambiental:
            </span>
            <Link
              className="py-2 px-4 m-1 text-white rounded-lg bg-green-400 shadow-lg block md:inline-block"
              to={`/humidityparameter/create/${id}`}
            >
              Agregar un parámetro de humedad ambiental
            </Link>
          </div>
        </>
      )}

      {!data.plant?.soil_moisture_parameter && (
        <>
          <div className="mb-4">
            <span className="py-4 px-4 text-gray-700 font-bold">
              Agrega un parámetro de Humedad de suelo:
            </span>
            <Link
              className="py-2 px-4 m-1 text-white rounded-lg bg-green-400 shadow-lg block md:inline-block"
              to={`/soilmoistureparameter/create/${id}`}
            >
              Agregar un parámetro de humedad de suelo
            </Link>
          </div>
        </>
      )}
      <div>
        {data.plant.temperature_parameters.map((params, index) => (
          <ParameterTemperature temperature={params} key={index} />
        ))}

        <ParameterHumidity humidity={data.plant.humidity_parameter} />

        <ParameterSoilMoisture
          soilmoisture={data.plant.soil_moisture_parameter}
        />
      </div>
    </div>
  );
}
```

Evidencia XXXIII: Gestionar Parámetros – Listar parámetros

En ***la Evidencia XXXIII: Gestionar Parámetros – Listar parámetros*** los parámetros se listan por cada planta, una planta puede tener más de un parámetro de temperatura, sin embargo, la humedad de suelo y ambiental corresponden a un solo parámetro cada cual, por lo mismo es que si en caso ya existen parámetros registrados de estos últimos dos los botones de agregar no aparecerán.

```
const { id } = useParams();
  const { register, handleSubmit, setValue } = useForm();
  const navigate = useNavigate();
  const location = useLocation();
  const createForm = location.pathname.includes("create");

  useQuery(GET_HUMIDITYPARAMETER, {
    variables: { id: Number(id) },
    skip: createForm,
    onCompleted: (data) => {
      setValue("max_value", data.humidity_parameter.max_value);
      setValue("description", data.humidity_parameter.description);
    },
    update(cache, { data: { create_humidity_parameter } }) {
      cache.modify({
        fields: {
          plants(existingHumidityParameters = []) {
            const newHumidtyParameterRef = cache.writeFragment({
              data: create_humidity_parameter,
              fragment: gql`
                fragment NewHumidityParameter on HumidityParameter {
                  id
                  type
                }
              `,
            });
            return [...existingHumidityParameters, newHumidtyParameterRef];
          },
        },
      });
    },
  });

  const [mutation] = useMutation(
    createForm
      ? HUMIDITYPARAMETER_CREATE_MUTATION
      : HUMIDITYPARAMETER_UPDATE_MUTATION
  );

  const executeMutation = async (data) => {
    const variables = {
      humidity: {
        ...data,
        max_value: Number(data.max_value),
        plant_id: Number(id),
      },
    };
    if (!createForm) {
      variables.id = Number(id);
      delete variables.humidity.plant_id;
    }
    try {
      const response = await mutation({
        variables,
      });
      if (response.data) {
        navigate(
          `/parameters/${
            response.data.create_humidity_parameter?.plant_id ||
            response.data.update_humidity_parameter?.plant_id
          }`,
          {
            replace: true,
          }
        );
      }
    } catch (error) {
      console.error(error);
    }
  };
```

Evidencia XXXIV: Gestionar Parámetros – Registro y modificación de parámetro de humedad ambiental

En la ***Evidencia XXXIV: Gestionar Parámetros – Registro y modificación de parámetro de humedad ambiental*** al igual que para los otros formularios presentados, el registro o actualización de la fase de una planta requiere verificar si es que se está pasando un parámetro, en este caso el id del parámetro y que la referencia contenga la palabra "***create***", en caso sea así se realiza una el registro, caso contrario se realiza la actualización, colocando previamente los valores del parámetro dentro de los inputs.

```
const { id } = useParams();
 const { register, handleSubmit, setValue } = useForm();
 const navigate = useNavigate();
 const location = useLocation();
 const createForm = location.pathname.includes("create");

 useQuery(GET_SOILMOISTUREPARAMETER, {
  variables: { id: Number(id) },
  skip: createForm,
  onCompleted: (data) => {
   setValue("min_value", data.soil_moisture_parameter.min_value);
   setValue("max_value", data.soil_moisture_parameter.max_value);
   setValue("description", data.soil_moisture_parameter.description);
  },
  update(cache, { data: { create_soil_moisture_parameter } }) {
   cache.modify({
    fields: {
     plants(existingSoilMoistureParameters = []) {
      const newSoilmoistureParameterRef = cache.writeFragment({
       data: create_soil_moisture_parameter,
       fragment: gql`
        fragment NewSoilMoistureParameter on SoilMoistureParameter {
         id
         type
        }
       `,
      });
      return [
       ...existingSoilMoistureParameters,
       newSoilmoistureParameterRef,
      ];
     },
    },
   });
  },
 });

 const [mutation] = useMutation(
  createForm
   ? SOILMOISTUREPARAMETER_CREATE_MUTATION
   : SOILMOISTUREPARAMETER_UPDATE_MUTATION
 );

 const executeMutation = async (data) => {
  const variables = {
   soilmoisture: {
    ...data,
    max_value: Number(data.max_value),
    min_value: Number(data.min_value),
    plant_id: Number(id),
   },
  };
  if (!createForm) {
   variables.id = Number(id);
   delete variables.soilmoisture.plant_id;
  }
  try {
   const response = await mutation({
    variables,
   });
   if (response.data) {
    navigate(
     `/parameters/${
      response.data.create_soil_moisture_parameter?.plant_id ||
      response.data.update_soil_moisture_parameter?.plant_id
     }`,
     {
      replace: true,
     }
    );
   }
  } catch (error) {
   console.error(error);
  }
 };
```

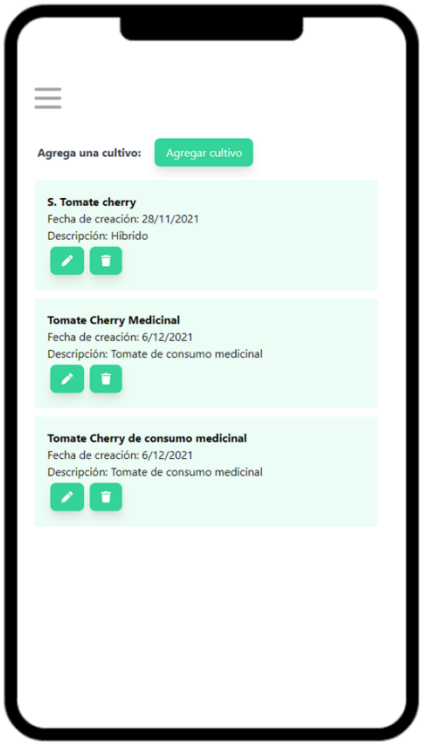
Evidencia XXXV: Gestionar Parámetros – Registro y modificación de parámetro de humedad de suelo

En la ***Evidencia XXXV: Gestionar Parámetros – Registro y modificación de parámetro de humedad de suelo*** se puede apreciar para poder realizar el registro o actualización del parámetro de humedad de suelo de una planta es necesario verificar si es que se está pasando un parámetro, en este caso el id del parámetro y que la referencia contenga la palabra "***create***", en caso sea así se realiza una el registro, caso contrario se realiza la actualización, colocando previamente los valores del parámetro dentro de los inputs.

```
const { id } = useParams();
  const { register, handleSubmit, setValue } = useForm();
  const navigate = useNavigate();
  const location = useLocation();
  const createForm = location.pathname.includes("create");

  useQuery(GET_TEMPERATUREPARAMETER, {
    variables: { id: Number(id) },
    skip: createForm,
    onCompleted: (data) => {
      setValue("description", data.temperature_parameter.description);
      setValue("min_value", data.temperature_parameter.min_value);
      setValue("max_value", data.temperature_parameter.max_value);
      setValue("start_hour", data.temperature_parameter.start_hour);
      setValue("end_hour", data.temperature_parameter.end_hour);
    },
    update(cache, { data: { create_temperature_parameter } }) {
      cache.modify({
        fields: {
          temperatureparameter(existingTemperatureParameters = []) {
            const newTemperatureParameterRef = cache.writeFragment({
              data: create_temperature_parameter,
              fragment: gql`
                fragment NewTemperatureParameter on TemperatureParameter {
                  id
                  type
                }
              `,
            });
            return [
              ...existingTemperatureParameters,
              newTemperatureParameterRef,
            ];
          },
        },
      });
    },
  });

  const [mutation] = useMutation(
    createForm
      ? TEMPERATUREPARAMETER_CREATE_MUTATION
      : TEMPERATUREPARAMETER_UPDATE_MUTATION
  );

  const executeMutation = async (data) => {
    const variables = {
      temperature: {
        ...data,
        min_value: Number(data.min_value),
        max_value: Number(data.max_value),
        plant_id: Number(id),
      },
    };
    if (!createForm) {
      variables.id = Number(id);
      delete variables.temperature.plant_id;
    }
    try {
      const response = await mutation({
        variables,
      });
      if (response.data) {
        navigate(
          `/parameters/${
            response.data.create_temperature_parameter?.plant_id ||
            response.data.update_temperature_parameter?.plant_id
          }`,
          {
            replace: true,
          }
        );
      }
    } catch (error) {
      console.error(error);
    }
  };
```

Evidencia XXXVI: Gestionar Parámetros – Registro y modificación de parámetro de temperatura

En la ***Evidencia XXXVI: Gestionar Parámetros – Registro y modificación de parámetro de temperatura*** se puede apreciar para poder realizar el registro o actualización del parámetro de temperatura de una planta es necesario verificar si es que se está pasando un parámetro, en este caso el id del parámetro y que la referencia contenga la palabra “create”, en caso sea así se realiza una el registro, caso contrario se realiza la actualización, colocando previamente los valores del parámetro dentro de los inputs.



Evidencia XXXVII: Desarrollo de la funcionalidad de gestionar cultivo – Interfaz listar cultivo

Evidencia XXXVIII: Desarrollo de la funcionalidad de gestionar cultivo – Interfaz registro y modificación de un cultivo

```
export default function Crop() {
  const { loading, error, data } = useQuery(CROP_QUERY);

  if (loading) return <p>Cargando……..……..</p>;
  if (error) return <p>Hay errores: {error.message}</p>;
  return (
    <div className="flex-grow bg-white">
      <br />
      <div className="mb-4">
        <span className="py-4 px-4 text-gray-700 font-bold">
          Agrega una cultivo:
        </span>
        <Link
          className="py-2 px-4 m-1 text-white rounded-lg bg-green-400 shadow-lg block md:inline-block"
          to="/crop/create"
        >
          Agregar cultivo
        </Link>
      </div>
      {data.crops.map((crop) => (
        <CropItem crop={crop} key={crop.id} />
      ))}
    </div>
  );
}
```

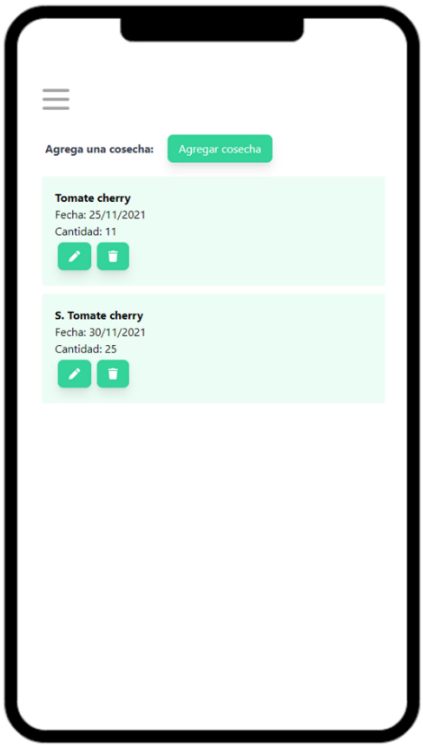
Evidencia XXXIX: Desarrollo de la funcionalidad de gestionar cultivo – Listar cultivo

En la ***Evidencia XXXIX: Desarrollo de la funcionalidad de gestionar cultivo – Listar cultivo*** se puede observar mediante la ***query*** se obtiene los datos de los cultivos, para mostrarlos estos se mapean y luego de ello se llama al componente ***CropItem***, ese corresponde a los ítems que se verán al momento de listar.

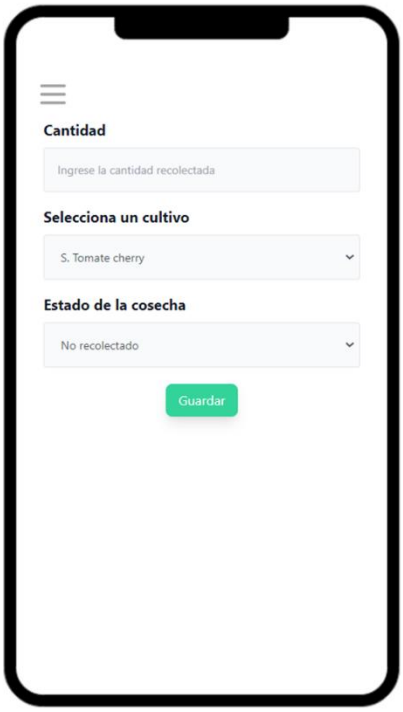
```
const [selectedPlant, setSelectedPlant] = React.useState(0);
  const { id } = useParams();
  const navigate = useNavigate();
  const form = React.useRef(null);

  useQuery(GET_CROP, {
    variables: { id: Number(id) },
    skip: !id,
    update(cache, { data: { create_crop } }) {
      cache.modify({
        fields: {
          plants(existingCrops = []) {
            const newCropRef = cache.writeFragment({
              data: create_plant,
              fragment: gql`
                fragment NewCrop on Crop {
                  id
                  type
                }
              `,
            });
            return [...existingCrops, newCropRef];
          },
        },
      });
    },
  });
  const { loading, error, data } = useQuery(PLANTS_QUERY);

  const [mutation] = useMutation(
    !id ? CREATE_CROP_MUTATION : UPDATE_CROP_MUTATION
  );

  if (loading) return <p>Cargando…… </p>;
  if (error) return <p>Hay errores: {error.message}</p>;

  const handleSubmit = (e, fn) => {
    e.preventDefault();
    const formData = new FormData(form.current);
    const formDataJson = Object.fromEntries(formData);
    fn(formDataJson);
  };
  const executeMutation = async (data) => {
    const variables = {
      data: {
        ...data,
        planted_area: Number(data.planted_area),
        plant_id: Number(data.plant_id),
        stageId: Number(data.stageId),
      },
    };
    try {
      const response = await mutation({
        variables,
      });
      if (response.data) navigate("/crop", { replace: true });
    } catch (error) {}
  };

  const findPlant = data.plants.find((plant) => plant.id === selectedPlant);
```

Evidencia XL: Desarrollo de la funcionalidad de gestionar cultivo – registro y modificación de un cultivo

En la ***Evidencia XL: Desarrollo de la funcionalidad de gestionar cultivo – registro y modificación de un cultivo*** debido a que los formularios de registrar y actualizar son el mismo, al igual que en los anteriores, primero se verifica si se ha pasado el parámetro id, en caso no sea así se realizara un registro con los calores ingresados, caso contrario se llenarán los campos y se permitirá actualizar los datos.



Evidencia XLI: Desarrollo de la funcionalidad de gestionar cosecha - Interfaz de listar cosechas



Evidencia XLII: Desarrollo de la funcionalidad de gestionar cosecha – Interfaz registro y modificación de un cultivo

```
export default function Harvest() {
 const { loading, error, data } = useQuery(HARVESTS_QUERY);

 if (loading) return <p>Cargando.....</p>;
 if (error) return <p>Hay errores: {error.message}</p>;

 return (
  <div className="flex-grow bg-white">
   <br />
   <div className="mb-4">
    <span className="py-4 px-4 text-gray-700 font-bold">
     Agrega una cosecha:
    </span>
    <Link
     className="py-2 px-4 m-1 text-white rounded-lg bg-green-400 shadow-lg block md:inline-block"
     to="/harvest/create"
    >
     Agregar cosecha
    </Link>
   </div>
   {data.harvests.map((harvest) => (
    <HarvestItem harvest={harvest} key={harvest.id} />
   ))}
  </div>
 );
}
```

Evidencia XLIII: Desarrollo de la funcionalidad de gestionar cosecha - Listar cosechas

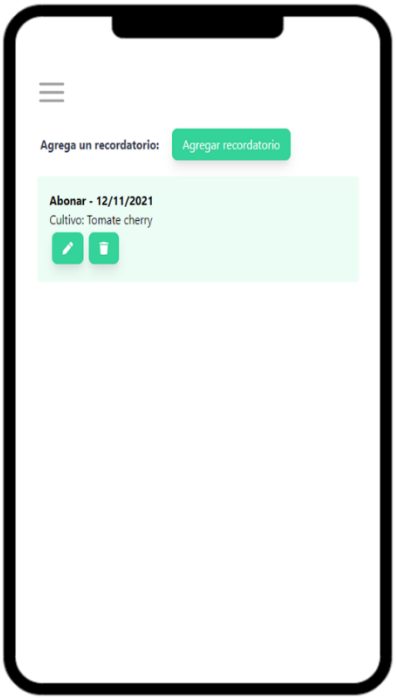
En la ***Evidencia XLIII: Desarrollo de la funcionalidad de gestionar cosecha - Listar cosechas*** se puede observar mediante la query se obtiene los datos de las cosechas registradas, para mostrarlos estos se mapean y luego de ello se llama al componente HarvestItem, ese corresponde a los ítems que se verán al momento de listar.

```
const { id } = useParams();
  const { register, handleSubmit, setValue } = useForm();
  const navigate = useNavigate();

  useQuery(HARVEST_QUERY, {
    variables: { id: Number(id) },
    skip: !id,
    onCompleted: (data) => {
      setValue("crop_id", data.harvest.crop_id);
      setValue("quantity", data.harvest.quantity);
      setValue("state", data.harvest.state);
    },
    update(cache, { data: { create_harvest } }) {
      cache.modify({
        fields: {
          plants(existingHarvest = []) {
            const newHarvestRef = cache.writeFragment({
              data: create_harvest,
              fragment: gql`
                fragment NewHarvest on Harvest {
                  id
                  type
                }
              `,
            });
            return [...existingHarvest, newHarvestRef];
          },
        },
      });
    },
  });
  const { loading, error, data } = useQuery(CROP_QUERY);

  const [mutation] = useMutation(
    !id ? HARVEST_CREATE_MUTATION : HARVEST_UPDATE_MUTATION
  );

  if (loading) return <p>Cargando.....</p>;
  if (error) return <p>Hay errores: {error.message}</p>;

  const executeMutation = async (data) => {
    const variables = {
      data: {
        ...data,
        crop_id: Number(data.crop_id),
        quantity: Number(data.quantity),
      },
    };
    if (id) {
      variables.id = Number(id);
      delete variables.data.crop_id;
    }
    try {
      const response = await mutation({
        variables,
      });
      if (response.data) navigate("/Harvest", { replace: true });
    } catch (error) {
      console.log(error);
    }
  };
```

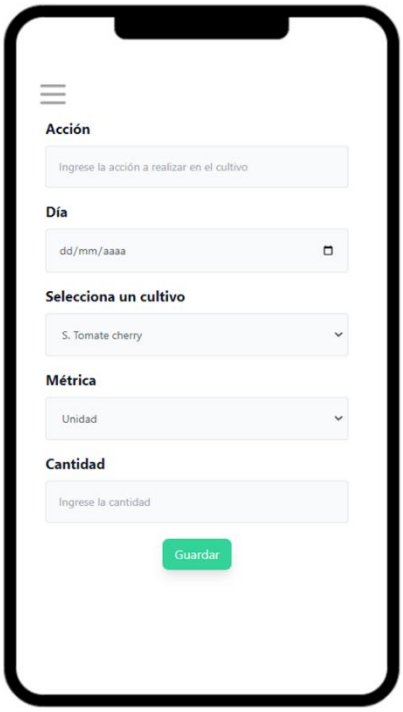
Evidencia XLIV: Desarrollo de la funcionalidad de gestionar cosecha – Registro y modificación de un cultivo

En la ***Evidencia XLIV: Desarrollo de la funcionalidad de gestionar cosecha – Registro y modificación de un cultivo*** debido a que los formularios de registrar y actualizar son el mismo, al igual que en los anteriores, primero se verifica si se ha pasado el parámetro id, en caso no sea así se realizara un registro con los calores ingresados, caso contrario se llenarán los campos y se permitirá actualizar los datos. Para las cosechas es necesario seleccionar a que cultivo le pertenece.

4.1.5. Sprint 5: Implementar recordatorios de eventos



Evidencia XLV: Desarrollo de la funcionalidad de los recordatorios de eventos – Interfaz de listar recordatorios



Evidencia *XLVI: Desarrollo de la funcionalidad de los recordatorios de eventos* – Interfaz *de registrar y actualizar recordatorios*

```
export default function Reminder() {
 const { loading, error, data } = useQuery(REMINDER_QUERY);

 if (loading) return <p>Cargando.....</p>;
 if (!data.reminders) {
  return (
   <div className="flex-grow bg-white">
    <br />
    <div className="mb-4">
     <span className="py-4 px-4 text-gray-700 font-bold">
      Agrega un recordatorio:
     </span>
     <Link
      className="py-2 px-4 m-1 text-white rounded-lg bg-green-400 shadow-lg block md:inline-block"
      to="/reminder/create"
     >
      Agregar recordatorio
     </Link>
    </div>
   </div>
  );
 }
 if (error) return <p>Hay errores: {error.message}</p>;
 return (
  <div className="flex-grow bg-white">
   <br />
   <div className="mb-4">
    <span className="py-4 px-4 text-gray-700 font-bold">
     Agrega un recordatorio:
    </span>
    <Link
     className="py-2 px-4 m-1 text-white rounded-lg bg-green-400 shadow-lg block md:inline-block"
     to="/reminder/create"
    >
     Agregar recordatorio
    </Link>
   </div>
   {data.reminders.map((reminder) => (
    <ReminderItem
     reminder={reminder}
     nickname={reminder.crop.nickname}
     key={reminder.id}
    />
   ))}
  </div>
 );
}
```

Evidencia XLVII: Desarrollo de la funcionalidad de los recordatorios de eventos – Listar recordatorios

En la ***Evidencia XLVII: Desarrollo de la funcionalidad de los recordatorios de eventos – Listar recordatorios*** se puede observar mediante la ***query*** se obtiene los datos de las cosechas registradas, para mostrarlos estos se mapean y luego de ello se llama al componente ReminderItem, ese corresponde a los ítems que se verán al momento de listar.

```
const { id } = useParams();
 const { register, handleSubmit, setValue } = useForm();
 const navigate = useNavigate();

 useQuery(REMINDER_QUERY, {
  variables: { id: Number(id) },
  skip: !id,
  onCompleted: (data) => {
   setValue("action", data.reminder.action);
   setValue("alarm_day", data.reminder.alarm_day);
   setValue("crop_id", data.reminder.crop_id);
   setValue("quantity", data.reminder.quantity);
   setValue("metrics", data.reminder.metrics);
   setValue("state", data.reminder.state);
  },
  update(cache, { data: { create_reminder } }) {
   cache.modify({
    fields: {
     plants(existingReminders = []) {
      const newReminderRef = cache.writeFragment({
       data: create_plant,
       fragment: gql`
        fragment NewReminder on Reminder {
         id
         type
        }
       `,
      });
      return [...existingReminders, newReminderRef];
     },
    },
   });
  },
 });
 const { loading, error, data } = useQuery(CROP_QUERY);

 const [mutation] = useMutation(
  !id ? REMINDER_CREATE_MUTATION : REMINDER_UPDATE_MUTATION
 );

 if (loading) return <p>Cargando.....</p>;
 if (error) return <p>Hay errores: {error.message}</p>;

 const executeMutation = async (data) => {
  const variables = {
   data: {
    ...data,
    quantity: Number(data.quantity),
    crop_id: Number(data.crop_id),
   },
  };
  if (id) {
   variables.id = Number(id);
   delete variables.data.crop_id;
  }
  try {
   const response = await mutation({
    variables,
   });
   if (response.data) navigate("/reminders", { replace: true });
  } catch (error) {}
 };
```

Evidencia *XLVIII: Desarrollo de la funcionalidad de los recordatorios de eventos* – Registrar y *actualizar recordatorios*

La ***Evidencia XLVIII: Desarrollo de la funcionalidad de los recordatorios de eventos – Registrar y actualizar recordatorios***, el cual funciona con la misma lógica que otros presentados, en caso no se haya pasado el parámetro id por la URL, significaría que lo que se desea realizar es un registro, caso contrario, se

colocaran los valores del recordatorio en los inputs para poder realizar una actualización.

4.1.6. Sprint 6: Implementar reportes

Evidencia XLIX: Desarrollo de la funcionalidad de los reportes – Interfaz de Reportes

```
const RECORDS_BY_MONTH = gql`
  query RecordsByMonth($month: Int!) {
    records_by_month(month: $month) {
      temperature
      humidity
      light
      soil_moisture
      records {
        temperature
        humidity
        soil_moisture
      }
    }
  }
`;

const options = {
  fill: true,
  animations: false,
  scales: {
    y: {
      min: 0,
    },
  },
  responsive: true,
  plugins: {
    legend: {
      display: true,
    },
  },
};

export default function Chart({ month }) {
  const month_1 = 4;
  const { data } = useQuery(RECORDS_BY_MONTH, { variables: { month_1 } });

  const variables = [
    data?.records_by_month.temperature,
    data?.records_by_month.humidity,
    69.9,
    data?.records_by_month.soil_moisture,
  ];
  const labels = [
    "Temperatura",
    "Humedad Ambiental",
    "Humedad de Suelo",
  ];

  const data_var = useMemo(function () {
    return {
      datasets: [
        {
          label: "Promedio de valores de las variables ambientales",
          tension: 0.3,
          data: variables,
          borderColor: "rgb(75, 192, 192)",
          backgroundColor: "rgba(75, 192, 192, 0.3)",
        },
      ],
      labels,
    };
  }, []);

  return (
    <div className="App">
      <Bar data={data_var} options={options} />
    </div>
  );
```

Evidencia L: Desarrollo de la funcionalidad de los reportes

La ***Evidencia L: Desarrollo de la funcionalidad de los reportes***, muestra cómo se trabajó la programación de los reportes, en este caso se hizo uso de Chart.js, la cual es una biblioteca de gráficos para JavaScript, colocando los datos de las variables cuantitativas como son la temperatura, humedad ambiental y humedad de suelo.

Entrando a la discusión de los resultados, con respecto al objetivo específico 1 planteado en la investigación de "Definir los componentes eléctricos y electrónicos a utilizar en el circuito de la Smart Greenhouse", este se desarrolló durante el Sprint 1, en cual se culminó con una selección de los componentes en base a criterios de selección [49] tomando en cuenta las principales variables de las que se habla en otras investigaciones [9] [10] [8], las cuales son la temperatura, la humedad ambiental, la humedad de suelo, la luminosidad y por último se añadió el sensor de lluvia, para poder apoyar la lógica del accionar del servo motor de la ventana que dio apertura y cierre a la ventana, sin embargo, a diferencia de otra de las investigaciones [6], la cual trabajo con un humidificador como actuador para regular la humedad ambiental hasta el valor esperado en caso esta bajara, para la presente investigación no se tomó en cuenta este actuador debido a que durante las pruebas que se realizaron haciendo uso del sensor de humedad, se detectó que esta no bajaba demasiado sino que regularmente se encontraba entre el 60% u 85% durante el todo el día y el valor que se espera para la humedad según lo que se conoce es no mayor al 70% [4], lo cual se debe a que si esta es demasiado alta podría ocasionar hongos en la planta de tomate sea cual sea la especie [2].

Para el desarrollo del aplicativo móvil, el cual forma parte del objetivo específico 2 de "Implementar una aplicación móvil para el control del ambiente y monitoreo en el cultivo de tomates nativos en la Smart Greenhouse" se trabajó con el Framework de React por lo cual la aplicación móvil o app es de tipo hibrida [30], para la parte de consultas a la base de datos se hizo uso de Graphql, y por el lado del hardware con Arduino y el módulo de WIFI ESP32 el cual permitió el envío de los datos a la base de datos, haciendo uso de un Router y manteniendo una conexión estable para que no haya inconvenientes, luego mediante una suscripción en Graphql se pudieran obtener los datos en tiempo real y los demás componentes ya mencionados en el primer Sprint. A partir del segundo Sprint hasta el sexto Sprint se desarrolló la programación del aplicativo para que el usuario pueda interactuar con el mismo de tal forma que se logró la gestión de todos los módulos establecidos con sus respectivas funcionalidades, el usuario podía apreciar en el menú principal los datos de las variables ambientales en tiempo real y además de ello, podía registrar los datos de las plantas, los cultivos, cosechas, recordatorios de eventos, estado de los actuadores y también podía apreciar mediante el apartado de reportes una gráfica del promedio mensual de las variables cuantitativas que se tienen como es la temperatura, la humedad ambiental y la humedad de suelo. A diferencia de otra de las investigaciones consideradas dentro de los antecedentes, en la cual el aplicativo se centraba más en solo mostrar el monitoreo y el control, se tomó en cuenta el registro de los datos ya mencionados para un mejor control del cultivo por parte del usuario.

Al momento de evaluar las funcionalidades del aplicativo se hizo uso de las pruebas de caja blanca y caja negra para la verificación y validación del aplicativo móvil, esto se puede apreciar en los ***Anexos N°8.***

Luego de la implementación, se pasó a hacer uso de la solución con un cultivo de tomates del tipo Solanum lycopersicum var. Cerasiforme, para poder llevar a cabo el objetivo específico 3 el cual es "Validar la implementación del Smart greenhouse para la determinación de su efectividad en el proceso óptimo del cultivo de las plantas de tomate nativo", para ello se contempló el desarrollo de otro cultivo de la misma especie en el invernadero, sin embargo, este no conto con el control, ni monitoreo por parte del sistema, la muestra para ambos fue de 5 ejemplares. Ambos cultivos se desarrollaron en un periodo aproximado de 3 meses, sin incidencias en plagas o enfermedades durante ese periodo, lo cual se tiene registrado en el ***Anexo N°2***, la cual se basa en un registro de cultivo de hortalizas [50].
En las siguientes figuras en las cuales el cultivo A (grupo experimental) es el que se desarrolló haciendo uso del sistema y el cultivo B (grupo control) el que no:

Figura XXXI: Cultivo B

Figura XXXII: Cultivo A

Figura XXXIII: Cultivo A

Figura XXXIV: Cultivo B

Como se puede apreciar ambos se desarrollaron de manera sana, pero al momento de rendir frutos el Cultivo A dio de 25 frutos con un peso de 1.2 kg y el Cultivo B dio la misma cantidad de 1.150 kg, tomando en cuenta que era la primera floración y frutos de ambos cultivos, los cuales no recibieron ningún tipo de químico al momento de ser abonados, es decir que todo el proceso fue orgánico.

Además, a diferencia del cultivo A, el cual posterior a dar sus frutos, perdió follaje y estaba empezando a salirle nuevo, un par de las plantas del cultivo B comenzaron a perder follaje y vitalidad como se muestra en la siguiente imagen:

Figura XXXV: Cultivo B

Tomando en cuenta los datos que se obtuvieron durante el uso de la solución se evaluó el rendimiento en base al peso de frutos cosechados por cada cultivo. Para determinar que esta métrica era la adecuada se hizo la revisión de otra investigación que trabajó con un cultivo de tomates [51], sin embargo, a diferencia de esta, que aplico su métrica en base a productos que había utilizado en el cultivo, en este proyecto aplica una solución IoT, por lo cual se adaptó el indicador a esta investigación:

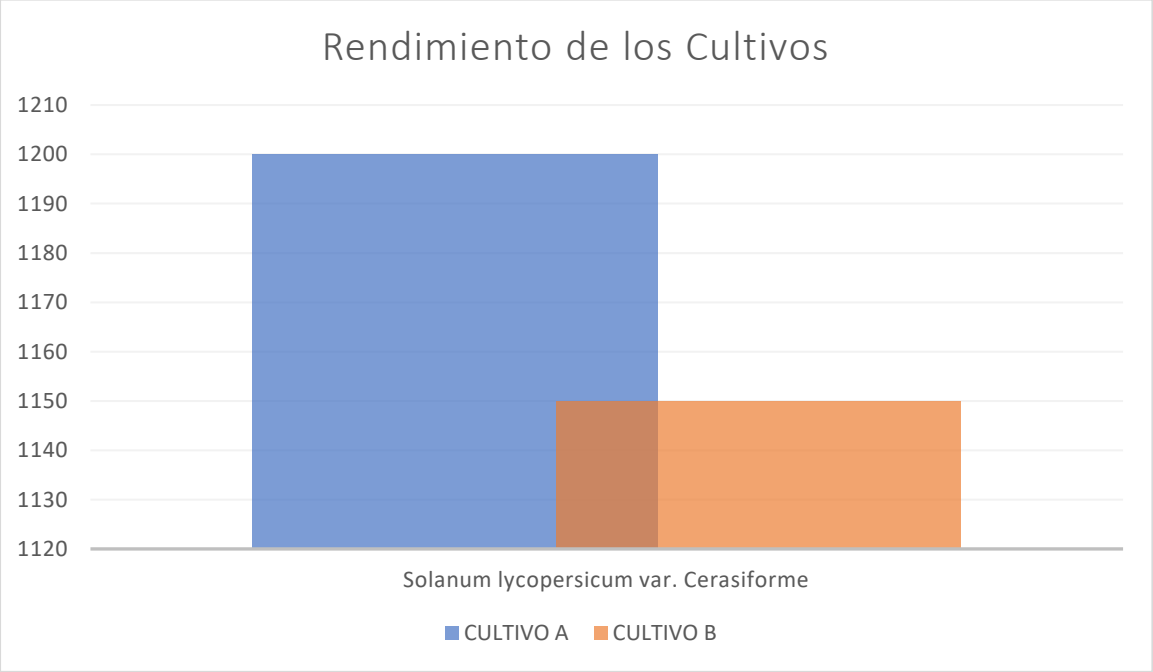


Gráfico I: Rendimiento de los cultivos

Con los valores obtenidos se puede calcular que el Cultivo A rindió un aproximado del 4,17% más que el cultivo B.

Para continuar con la investigación se plantó otro lote del mismo cultivo durante el mes de marzo del 2022, en la cual el cultivo se desarrolló desde la etapa de germinación, sin embargo, por falta de tiempo no se podrán demostrar los resultados en la presente investigación debido a que las plantas aún no pasan por la etapa de floración. Con respecto al los dos primeros cultivos, es decir el Cultivo A y Cultivo B aún se encuentran en un estado adecuado, sin embargo, como se mencionó anteriormente se perdieron dos plantas del cultivo B.

V. CONCLUSIONES

Se logró definir los componentes eléctricos y electrónicos necesarios para el sistema IoT, los cuales permitieron la obtención de los datos de las variables ambientales, así como el accionar de los actuadores.

Se implementó el aplicativo móvil que permitió el control del ambiente y monitoreo en la Smart Greenhouse para el cultivo de tomates nativos peruanos.

Al realizarse la validación del rendimiento del cultivo se comprobó que el grupo experimental tuvo mayor rendimiento al del grupo control.

VI. RECOMENDACIONES

En la presente investigación se llegó a probar el sistema IoT desarrollado en el cultivo de Solanum lycopersicum var. Cerasiforme, por lo cual se recomienda ampliar la investigación considerando otras variedades de tomates nativos peruanos.

Se recomienda para otras investigaciones incluir la automatización del riego por goteo, para independizar más al cultivo, disminuyéndole carga laboral al cultivador.

Para disminuir el impacto ambiental y los gastos de consumo de energía eléctrica que alimentan el circuito del Sistema IoT, se recomienda complementar la investigación con la implementación de un proyecto de energías renovables.

REFERENCIAS

[1] Viceministerio de Desarrollo Estratégico de los Recursos Naturales, DIRECCIÓN GENERAL DE DIVERSIDAD BIOLÓGICA, DIRECCIÓN GENERAL DE DIVERSIDAD BIOLÓGICA, «SERVICIO DE CONSULTORIA PARA LA ELABORACIÓN DE LA LINEA DE BASE DE LA DIVERSIDAD GENÉTICA DEL TOMATE NATIVO: PROSPECCION DE LA DIVERSIDAD, ESTUDIO SOCIOECONÓMICO, ECOLÓGICO, DE ORGANISMOS Y MICROORGANISMOS, FLUJO DE GENES Y SISTEMATIZACIÓN,» Ministerio del ambiente, Lima, 2019.

[2] Ministerio del Ambiente, Viceministerio de Desarrollo Estratégico de los Recursos Naturales, Dirección General de Diversidad Biológica, Dirección de Recursos GenéϴcosyBioseguridad, «Línea de base de la diversidad del tomate peruano con fines de bioseguridad,» Ministerio del Ambiente, Viceministerio de Desarrollo Estratégico de los Recursos Naturales, Dirección General de Diversidad Biológica, Dirección de Recursos GenéϴcosyBioseguridad, Lima, 2020.

[3] Ministerio del Ambiente, Viceministerio de Desarrollo Estratégico de los Recursos Naturales, Dirección General de Cambio Climático, Desertificación y Recursos Hídricos, Proyecto Tercera Comunicación Nacional de Cambio Climático, «El Perú y el Cambio Climático Tercera Comunicación Nacional del Perú a la Convención Marco de las Naciones Unidas sobre el Cambio Climático,» Ministerio del Ambiente, Lima, 2016.

[4] RED AGRICOLA, «RED AGRICOLA,» Marzo 2018. [En línea]. Available: https://www.redagricola.com/pe/reinventar-el-cultivo-del-tomate/. [Último acceso: Mayo 2021].

[5] ORACLE, «ORACLE,» ORACLE, [En línea]. Available: https://www.oracle.com/ar/internet-of-things/what-is-iot/. [Último acceso: 2 Mayo 2021].

[6] C. E. Reyna Humán, «Sistema Automatizado para el monitoreo y control de la humedad en un invernadero,» Pontificia Universidad Católica del Perú, Lima, 2015.

[7] G. Aliga Mendoza y P. R. Quispe Bolaños, «Sistema de control de humedad relativa para un invernadero utilizando el controlador lógico programable,» UNIVERSIDAD NACIONAL DE HUANCAVELICA, Huancavelica, 2015.

[8] J. C. Machaca Cutipa, «CONTROL PREDICTIVO MULTIVARIABLE Y SU EFICACIA EN LA OPTIMIZACIÓN DEL CLIMA DE UN INVERNADERO,» UNIVERSIDAD NACIONAL DE SAN AGUSTIN DE AREQUIPA, Arequipa, 2018.

[9] E. G. Cruz Vásquez y B. F. Lamadrid, «APLICACIÓN MÓVIL DE MONITORIZACIÓN Y CONTROL DE UN INVERNADERO DOMÉSTICO AUTOMATIZADO USANDO ARDUINO,» UNIVERSIDAD NACIONAL DE TRUJILLO, Trujillo, 2018.

[10] G. N. Gomez Pacci, «Diseño e implementación de un controlador difuso utilizando arduino para la autimatización de un mini invernadero de rosas,» UNIVERSIDAD RICARDO PALMA, Lima, 2019.

[11] J.-h. Hwang y H. Yoe, «Design of Wireless Sensor Network based Smart Greenhouse System,» *Conference Papers & Proceedings*, pp. 43-48, 2016.

[12] L. Jong Goo, J. Young Kyun , Y. Sung Wook , C. Man Kwon , . K. Hyeon Tae y Y. Yong Cheol , «Field Survey on Smart Greenhouse,» *Journal of Bio-Environment Control*, vol. 27, nº 2, pp. 166-172, 2018.

[13] IRJET Journa, «Monitoring of Smart Greenhouse,» *IRJET Journal,* vol. 3, nº 11, 2016.

[14] R. Kishore Kodali, V. Jain y S. Karagwal, «IoT based Smart Greenhouse,» Department of Electronics and Communication Engineering National Institute of Technology, Warangal, 2017.

[15] Pradyumna K. Tripathy, Ajaya K. Tripathy, Aditi Agarwal y Saraju P. Mohanty, «MyGreen: An IoT-Enabled Smart Greenhouse for Sustainable Agriculture,» *IEEE Consumer Electronics Magazine,* vol. 10, nº 4, p. 62, 2021.

[16] NOVAGRIC, «NOVAGRIC,» Novedades Agrícolas S.A, 2016. [En línea]. Available: https://www.novagric.com/es/invernaderos-tomate. [Último acceso: 04 Junio 2020].

[17] Red Hat, «Red Hat,» Red Hat, 8 Enero 2019. [En línea]. Available: https://www.redhat.com/es/topics/internet-of-things/what-is-iot. [Último acceso: 15 Mayo 2021].

[18] ARDUINO, «ARDUINO,» 2021. [En línea]. Available: https://www.arduino.cc/en/Main/Products. [Último acceso: 17 Mayo 2021].

[19] J. C. Herrero Herranz y J. Sánchez Allende, «Una mirada al mundo arduino,» *Revista,* vol. 8, p. 28, 2015.

[20] ESPRESSIF, «ESPRESSIF,» 2020. [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/protocols/esp_websocket_client.html. [Último acceso: 2022 04 01].

[21] G. Maloy Smith, «DEWEsoft,» 09 Marzo 2020. [En línea]. Available: https://dewesoft.com/es/daq/que-es-un-sensor. [Último acceso: 15 Mayo 2021].

[22] GENERA CREATIVE GROUP, «AUTYCOM,» AUTYCOM, 15 Octubre 2021. [En línea]. Available: https://www.autycom.com/que-es-un-actuador/. [Último acceso: 15 Mayo 2021].

[23] DESIGNING BUILDINGS, «DESIGNING BUILDINGS,» 4 Noviembre 2020. [En línea]. Available: https://www.designingbuildings.co.uk/wiki/Smart_greenhouse#:~:text=The%20smart%20greenhouse%20is%20a,and%20automate%20the%20growing%20process.. [Último acceso: 15 Mayo 2021].

[24] «Environmental monitoring and disease detection of plants in smart,» *Journal of Physics Comunications,* vol. 4, p. 15, 2020.

[25] X. Wang y H. Yu, «Research on Control System of Intelligent Greenhouse of IoT,» *Journal of Physics: Conference Series,* p. 8, 2019.

[26] C. O. Sereati , H. Sutrisno y S. D. Putra, «Design of IoT Monitoring System Based on LoRaWAN,» *2nd International Conference on Agriculture and Applied Science (ICoAAS 2021),* vol. 2, p. 8, 2021.

[27] Heri, R. K. Nasution, B. C. Guptan, D. K. O. Larosa, D. Sofhani, H. Siagian, D. Perangin-angin y E. Dodi Suryanto, «Design of monitoring and automation systems for greenhouse,» *TALENTA CEST II 2019,* p. 7, 2020.

[28] H. JINDAL, A. KAUR , ARSHITA, S. KUMAR , N. GAUTAM y R. KUMAR, «IoT BASED SMART AGRICULTURE: A STUDY,» Punjab, 2021.

[29] V. Khanh Quy, N. Van Hau, D. Van Anh, N. Minh Quy, N. Tien Ban, S. Lanza, G. Randazzo y A. Muzirafuti, «IoT-Enabled Smart Agriculture: Architecture, Applications,,» Messina, 2022.

[30] Adapptative , «Adapptative,» 18 Abril 2018. [En línea]. Available: https://adapptative.com/tipos-de-aplicaciones-moviles-o-apps/. [Último acceso: 15 Mayo 2021].

[31] Android, «Android,» 2021. [En línea]. Available: https://www.android.com/intl/es_es/. [Último acceso: 17 Mayo 2021].

[32] React, «React,» 2021. [En línea]. Available: https://es.reactjs.org/. [Último acceso: 2021].

[33] Visual Studio Code, «Visual Studio Code,» [En línea]. Available: https://code.visualstudio.com/. [Último acceso: 15 Mayo 2021].

[34] Microsoft, «Microsoft,» Microsoft, 12 Febrero 2018. [En línea]. Available: https://www.oracle.com/ar/database/what-is-a-relational-database/. [Último acceso: 04 Junio 2021].

[35] GraphQL, «GraphQL,» 2021. [En línea]. Available: https://graphql.org/learn/. [Último acceso: 2021].

[36] Docker, «Docker,» 2021. [En línea]. Available: https://www.docker.com/why-docker. [Último acceso: 2021].

[37] J. C. Salazar, T. Álvaro, J. C. Linares, A. Lozano y L. Valbuena, «Scrum versus XP: similitudes y diferencias,» Universidad Distrital Francisco José de Caldas, Bogotá, 2018.

[38] S. Merzouk, A. Cherkaoui, A. Marzak y S. Nawal, «IoT methodologies: comparative study,» Sidi Othman, 2020.

[39] J. Sutherland y K. Schwaber, «La Guía de Scrum,» 2020.

[40] Sensirion, «Datasheet SHT3x-DIS,» Sensirion, 2016.

[41] Aosong(Guangzhou) Electronics Co, «Digital-output Digital-output Digital-output Digital-output relative relative relative relative humidity humidity humidity humidity & temperature temperature temperature temperature sensor/module sensor/module sensor/module sensor/module,» Aosong(Guangzhou) Electronics Co, Guangzhou.

[42] Texas Instruments, «LM35 Precision Centigrade Temperature Sensors,» Texas Instruments, Dallas, 2017.

[43] Maxim Integrated Products, «Programmable Resolution Wire Digital Thermometer,» Maxim Integrated Products, 2019.

[44] AOSONG, «Temperature and humidity module AM2301 Product Manual,» AOSONG.

[45] ARIAN, «Pt100, su operación, instalación y tablas,» ARIAN.

[46] «Naylamp Mecatronics,» Naylamp Mecatronics, [En línea]. Available: https://naylampmechatronics.com. [Último acceso: 18 Mayo 2021].

[47] Espressif Systems, «ESPRESSIF SMART CONNECTIVITY PLATFORM: ESP8266,» Espressif Systems, 2013.

[48] Espressif Systems, «ESP32 Series,» Espressif Systems, 2021.

[49] A. Brunete, P. San Segundo y R. Herrero, «Introducción a la Automatización Industrial,» Universidad Politécnica de Madrid, Madrid, 2020.

[50] J. H. B. Méndez, «Construyendo Huertos Caseros,» Georgia, 2022.

[5 W. . I. Lanchimba Sopalo, L. G. González Gómez, y T. Boicet Fabré,
1] «COMPORTAMIENTO DEL TOMATE (SOLANUM LYCOPERSICUM, L.)

VARIEDAD AMALIA EN CUBA Y ECUADOR AL APLICARLE QUITOMAX,» Cotopaxi, Granma, 2020.

[52] Kicad, «Kicad,» 2021. [En línea]. Available: https://www.kicad.org/about/kicad/. [Último acceso: 2021].

[53] R. A. d. l. l. Española, «Real Academia de la legua Española,» 2021. [En línea]. Available: https://dle.rae.es/invernadero. [Último acceso: 17 Mayo 2021].

[54] SwitchDoc Labs Blog, «Tutorial – Using Capacitive Soil Moisture Sensors on the Raspberry Pi,» SwitchDoc Labs Blog, 2020.

[55] Servisoftcorp, «Servisoftcorp,» 2010. [En línea]. Available: https://servisoftcorp.com/definicion-y-como-funcionan-las-aplicaciones-moviles/#Que_es_una_aplicacion_movil. [Último acceso: 17 Mayo 2021].

[56] SECMOTIC, «Servicio de desarrollo de sistemas IoT,» SECMOTIC INNOVATION, S.L., Sevilla, 2021.

[57] Texas Instruments, «LMx93-N, LM2903-N Low-Power, Low-Offset Voltage, Dual Comparators,» Texas Instruments, Dallas, 2018.

[58] ORACLE, «ORACLE,» ORACLE, 2018. [En línea]. Available: https://docs.oracle.com/es/solutions/spark-master-worker-mode/index.html#GUID-7A8A695C-1B18-4A7D-A0DD-35C4685BFEFF. [Último acceso: 2020].

[59] Components 101, «Components 101,» 2021. [En línea]. Available: https://components101.com/sensors/rain-drop-sensor-module. [Último acceso: 2021].

[60] Farm Structures Laboratory, Department of Natural Resources and Agricultural Engineering, «Application of Internet of Things (IoT) for Optimized,» Chrysoula Nikita-Martzopoulou, Athens, 2021.

[61] S. M. Velásquez, D. E. Monsalve Sossa, M. E. Zapata, M. E. Gómez Adasme y J. P. Ríos, «Pruebas a aplicaciones móviles: avances y retos,» Antioquia, 2018.

ANEXOS

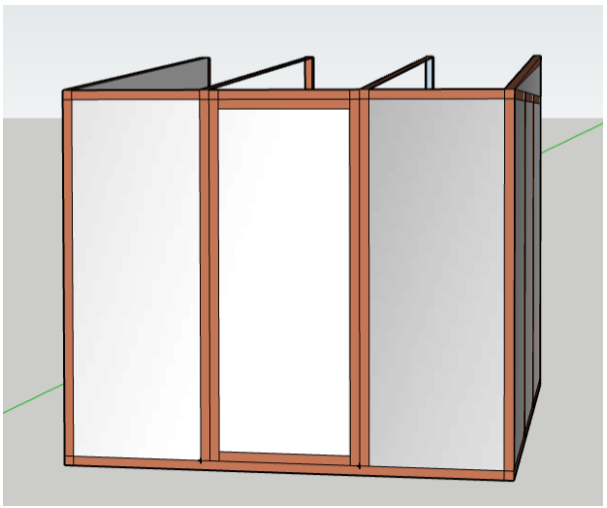
ANEXO N° 01: Cuestionario

Cuestionario Para la Entrevista	
Nombre del entrevistador:	
Nombre del entrevistado:	
Ocupación:	
Fecha de la entrevista:	
Preguntas del cuestionario	
1. ¿Qué conoce acerca de los cultivos de tomate nativo en el Perú?	
2. ¿Cómo se lleva a cabo actualmente el proceso de cultivo de tomate en Perú?	
3. ¿Cuáles son las consideraciones que se deben tener al tener un cultivo de tomate?	
4. ¿Qué tanto afecta el clima al cultivo de tomate?	
5. ¿Cuáles son las principales plagas que lo afectan al cultivo de tomate?	
6. ¿Cuáles son las principales enfermedades que afectan al cultivo de tomate?	
7. ¿Cuáles son las tecnologías que se utilizan en el Perú aplicadas en los cultivos de tomate?	
8. ¿A comparación de otros cultivos de hortalizas cual es el nivel de complejidad de un cultivo de tomate?	
9. ¿Qué recomendación daría al momento de aplicar una tecnología nueva para un cultivo de tomate nativo?	
10. ¿De qué fuentes recomienda buscar información acerca de cultivos nativos peruanos?	

ANEXO N° 02: Ficha de Registro de datos

Cultivo	Variedad	Fumigación		Fertilizante			Notas
		Cuando	¿Con qué?	Cuando	¿Con qué?	Cantidad	
A	Solanum lycopersicum var. Cerasiforme	15/01/22	Aceite de neem + Jabón potásico	15/01/22	Humus de Lombriz	2 kg mensual	Sin incidencias en plagas o enfermedades
		15/02/22		15/02/22			Sin incidencias en plagas o enfermedades
		15/03/22		15/03/22			Sin incidencias en plagas o enfermedades
		15/04/22		15/04/22			Sin incidencias en plagas o enfermedades
B		15/01/22	Aceite de neem + Jabón potásico	15/01/22	Humus de Lombriz	2 kg	Sin incidencias en plagas o enfermedades
		15/02/22		15/02/22			Sin incidencias en plagas o enfermedades
		15/03/22		15/03/22			Sin incidencias en plagas o enfermedades
		15/04/22		15/04/22			Sin incidencias en plagas o enfermedades

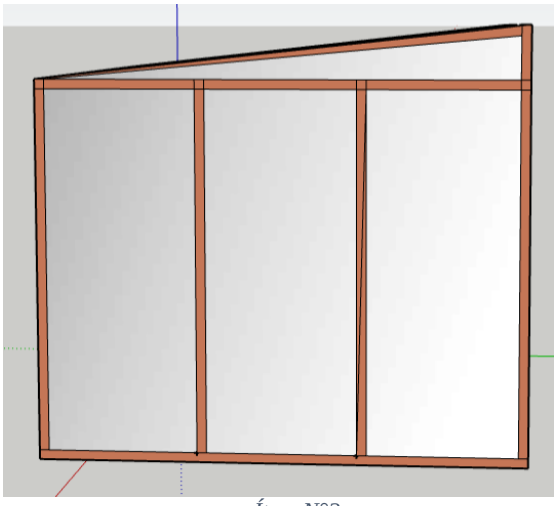
ANEXO N° 03: Diseño de la Greenhouse



Ítem N°1

El ***Ítem N°1*** representa el diseño de la parte frontal de la greenhouse del proyecto, teniendo como dimensiones las siguientes medidas:

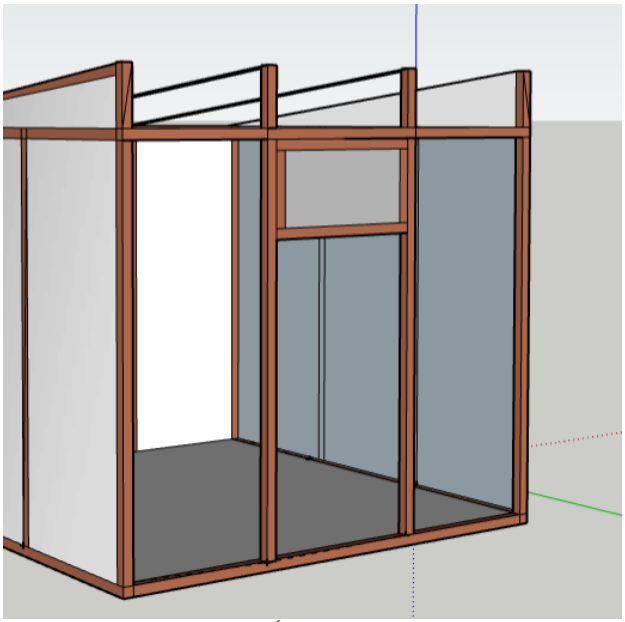
- ***Ancho total:*** 240 cm
- ***Alto total:*** 200 cm
- ***Paredes izquierda y derecha***: 190cm Largo x70 cm Ancho
- ***Puerta central:*** 180cm Largo x70 cm Ancho



Ítem N°2

El ***Ítem N°2*** representa el diseño de las partes laterales de la greenhouse del proyecto, teniendo como dimensiones las siguientes medidas:

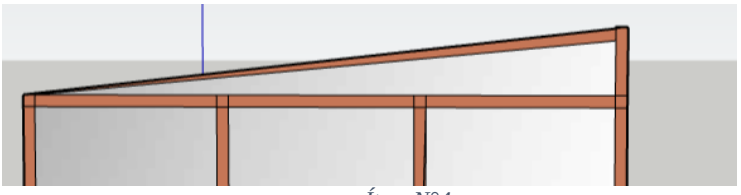
- ***Ancho total:*** 255 cm
- ***Alto total:*** 200 cm
- ***Paredes izquierda, centro y derecha***: 190cm Largo x78.3 cm Ancho



Ítem N°3

La ***Ítem N°3*** representa el diseño de la parte trasera de la greenhouse del proyecto, teniendo como dimensiones las siguientes medidas:

- ***Ancho total:*** 255 cm
- ***Alto total:*** 200 cm
- ***Paredes izquierda, centro y derecha***: 190cm Largo x78.3 cm Ancho

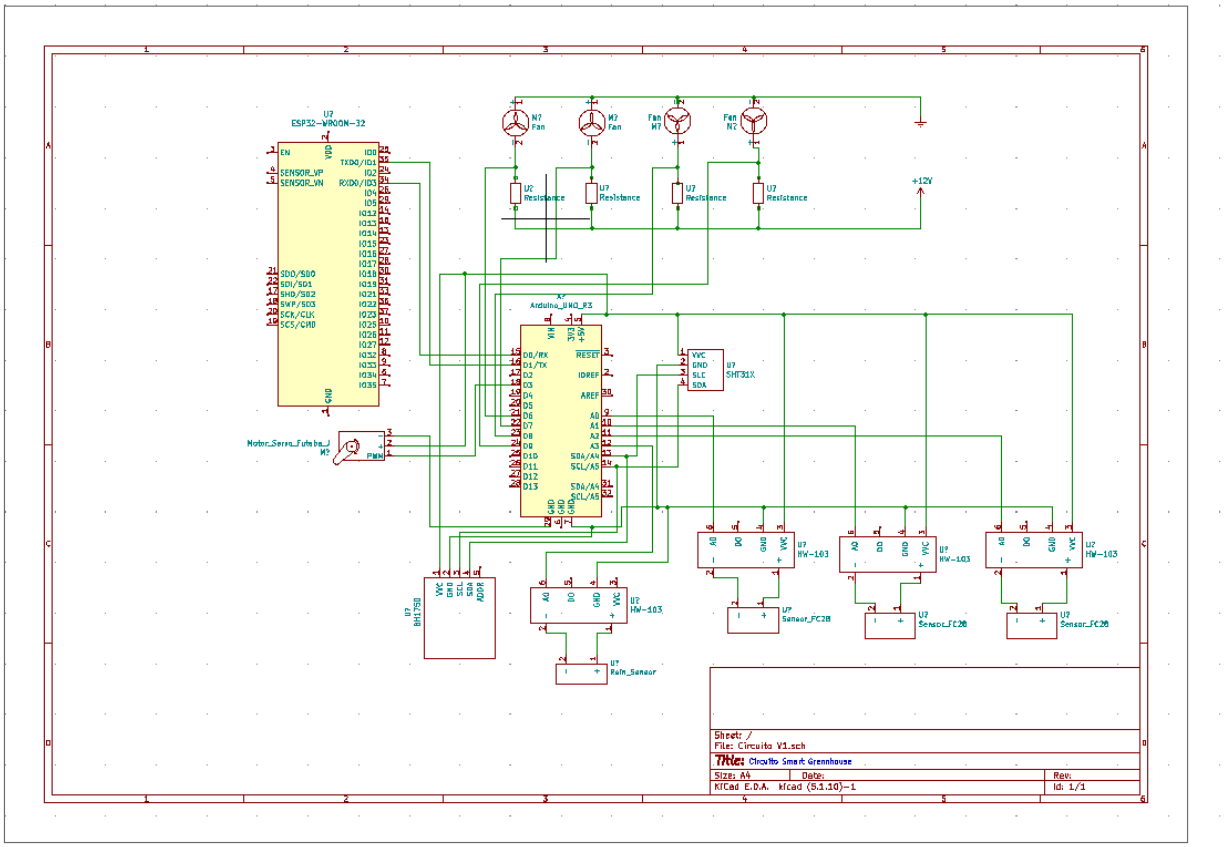


Ítem N°4

El ***Ítem N°4*** el diseño de la parte del techo de la greenhouse del proyecto, teniendo como dimensiones las siguientes medidas:

- ***Ancho total:*** 255 cm
- ***Alto total:*** 28 cm

ANEXO N° 04: Diseño del circuito



En el diseño del circuito de la Smart greenhouse, tomando en cuenta los componentes seleccionados en la Tarea I-3: Selección de los componentes eléctricos y electrónicos, este diseño fue desarrollado en la herramienta de Kicad [52], la cual es una aplicación que permite realizar diseños de circuitos, PCB's y componentes.

- **ESP32-WROOM-32:** El componente con este nombre hace referencia al módulo de WIFI ESP32, el cual se conecta de manera serial al Arduino por los pines RX y TX.
- **ARDUINO_UNO_R3:** El componente con este nombre hace referencia a la placa Arduino, la cual se encuentra conectada a los sensores, actuadores y al módulo WIFI.
- **SENSOR_FC28:** El componente con este nombre hace referencia a los sensores de humedad de suelo los cuales en este caso se encuentra conectado a la placa Arduino mediante los pines analógicos.
- **RAIN_SENSOR:** El componente con este nombre hace referencia al sensor de lluvia, el cual también se encuentra conectado a la placa Arduino mediante uno de los pines analógicos.

- **SHT31X:** El componente con este nombre hace referencia al sensor de temperatura y humedad relativa, el cual se encuentra conectado mediante el protocolo de conexión serial I2C a uno de los pines analógicos de la placa Arduino.
- **BH1750:** El componente con este nombre hace referencia al sensor de luminosidad, el cual de igual forma que el SHT31X se encuentra conectado mediante el protocolo de conexión serial I2C a uno de los pines analógicos de la placa Arduino.
- **MOTOR_SERVO_FUTABA:** El componente con este nombre hace referencia al motor servo de 11kg seleccionado, el cual se utilizó para la apertura y cierre de la ventana, este se encuentra conectado a la placa Arduino mediante uno de los pines digitales.
- **FAN:** Los componentes con este nombre hacen referencia a los ventiladores utilizados para ventilar el cultivo, en este caso estos ventiladores utilizan un voltaje de entre 6 y 12 voltios, estos se encuentran conectados a la placa Arduino mediante los pines digitales.
- **RESISTANCE:** El componente con este nombre hace referencia a las resistencias utilizadas para los ventiladores, para que así el flujo de corriente no dañe a los ventiladores, el valor de estas es de 220 Ω.

ANEXO N° 05: Construcción de la Greenhouse

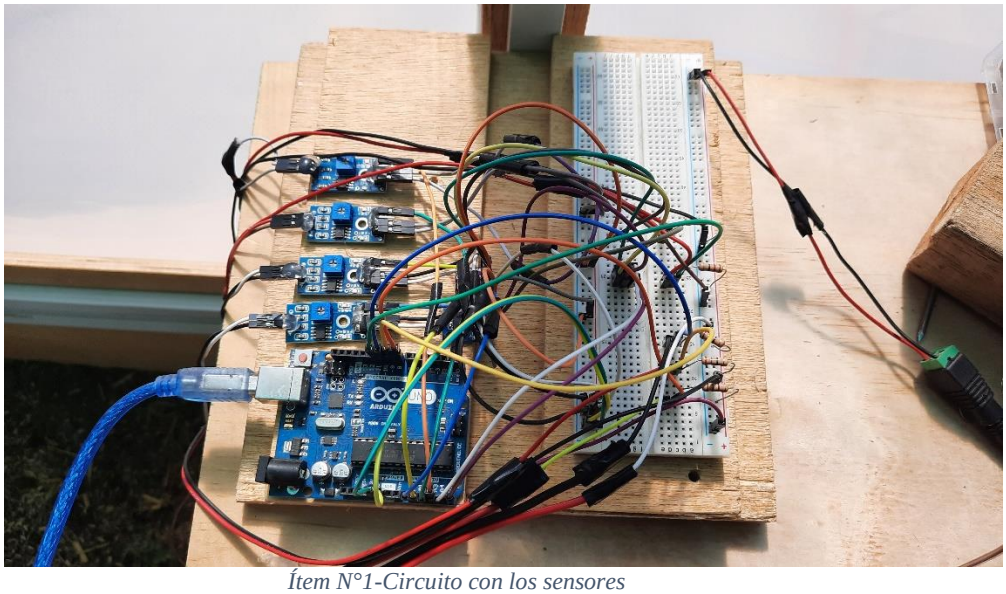
Ítem N°1

Ítem N°2

Ítem N°3

Ítem N°4

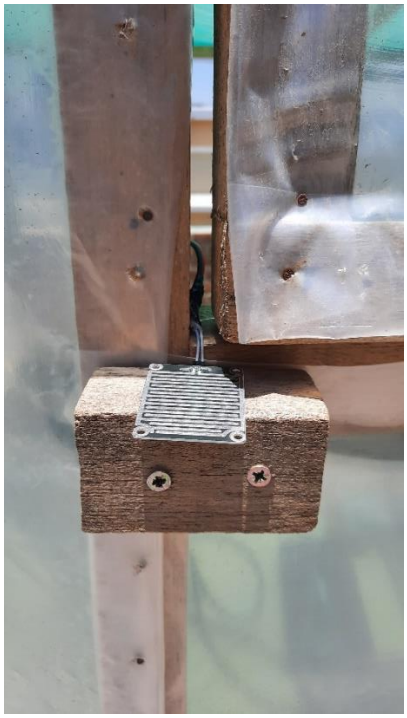
ANEXO N° 06: Implementación del circuito en la Greenhouse



Ítem N°1-Circuito con los sensores

Ítem N°2-Ventiladores

Ítem N°3– Servomotor



Ítem N°4-Sensor de lluvia

ANEXO N° 07: Cultivo en la Smart Greenhouse

Ítem N°1

Ítem N°2

Ítem N°3

Ítem N°4

ANEXO N° 08: Pruebas de Caja Blanca y Caja Negra

DESCRIPCIÓN DE PRUEBAS DE CAJA BLANCA		
Requisito		
Módulo / Área Funcional / Subproceso Implementación del circuito	**Tipo de requisito** Funcional	**Código del requisito** -
Descripción del requisito Realizar la implementación del circuito para la obtención de variables ambientales mediante sensores y control de actuadores dentro de la Greenhouse (invernadero).		
Caso de prueba		
Código de prueba PCN01	**Caso de prueba** Captura de datos de los sensores y funcionalidad de los actuadores	**Fecha de prueba** 10/03/2022
Funcionalidad / Característica a evaluar Implementar circuito	**Datos de entrada / Acciones de entrada**	**Resultado esperado** Mediante el monitor serie del IDE de Arduino se deben obtener los datos correspondientes a las variables ambientales, las cuales deben de ser obtenidas mediante los sensores y el funcionamiento de los actuadores controlados por la placa Arduino.
Requerimientos de ambiente de pruebas Hardware: circuito con los componentes IoT Software: programación realizada en Arduino uno		**Condiciones / Restricciones** Ninguna
Seguimiento		
Resultado obtenido Datos de las variables ambientales y funcionamiento de los actuadores.	**Estado actual** Conforme	**Observaciones** Podría mejorarse el circuito aumentando la cantidad de actuadores y la alimentación eléctrica del sistema.
Correcciones		
Fecha de cambio de estado	**Observaciones**	

Ítem N°1

DESCRIPCIÓN DE PRUEBAS DE CAJA NEGRA		
Requisito		
Módulo / Área Funcional / Subproceso IoT	**Tipo de requisito** -	**Código del requisito** -
Descripción del requisito Obtención de los datos correspondientes a las variables ambientales mediante los sensores y mostrados por el aplicativo móvil.		
Caso de prueba		
Código de prueba PCN02	**Caso de prueba** Obtención de los datos	**Fecha de prueba** 10/05/2022
Funcionalidad / Característica a evaluar Obtener los datos de los sensores de temperatura y humedad, lluvia, luminosidad y humedad de suelo mediante el aplicativo	**Datos de entrada / Acciones de entrada** Datos captados por lo sensores	**Resultado esperado** En el aplicativo se muestran los datos obtenidos por los sensores de temperatura y humedad, lluvia, luminosidad y humedad de suelo.
Requerimientos de ambiente de pruebas Hardware: circuito con los componentes IoT conectado al módulo de WIFI ESP32 Software: programación realizada en Arduino uno, en el servidor y en el aplicativo		**Condiciones / Restricciones** Ninguna.
Seguimiento		
Resultado obtenido El aplicativo se muestra los datos obtenidos por los sensores de temperatura y humedad, lluvia, luminosidad y humedad de suelo	**Estado actual** Conforme	**Observaciones** Ninguna
Correcciones		
Fecha de cambio de estado	**Observaciones**	

Ítem N°2

DESCRIPCIÓN DE PRUEBAS DE CAJA NEGRA		
Requisito		
Módulo / Área Funcional / Subproceso Planta	**Tipo de requisito** Funcional	**Código del requisito** -
Descripción del requisito El usuario debe poder gestionar la planta mediante el aplicativo.		
Caso de prueba		
Código de prueba PCN03	**Caso de prueba** Gestión de la Planta	**Fecha de prueba** 10/05/2022
Funcionalidad / Característica a evaluar Gestionar planta	**Datos de entrada / Acciones de entrada** Operaciones CRUD de la planta	**Resultado esperado** El sistema listará, buscará, agregará, modificará y eliminará una planta correctamente.
Requerimientos de ambiente de pruebas Hardware: circuito con los componentes IoT conectado al módulo de WIFI ESP32 Software: programación realizada en Arduino uno, en el servidor y en el aplicativo		**Condiciones / Restricciones** Ninguna.
Seguimiento		
Resultado obtenido Las operaciones CRUD de planta se han realizado correctamente.	**Estado actual** Conforme	**Observaciones** Al momento de eliminar falta especificarle al usuario que no puede eliminar una planta que contiene alguna relación con cultivo, parámetros o fase.
Correcciones		
Fecha de cambio de estado	**Observaciones**	

Ítem N°3

DESCRIPCIÓN DE PRUEBAS DE CAJA NEGRA		
Requisito		
Módulo / Área Funcional / Sub proceso Fase	**Tipo de requisito** Funcional	**Código del requisito** -
Descripción del requisito El usuario debe poder gestionar la fase de una planta mediante el aplicativo.		
Caso de prueba		
Código de prueba PCN03	**Caso de prueba** Gestión de la fase	**Fecha de prueba** 10/05/2022
Funcionalidad / Característica a evaluar Gestionar fase	**Datos de entrada / Acciones de entrada** Operaciones CRUD de la fase	**Resultado esperado** El sistema listará, buscará, agregará, modificará y eliminará una fase de una planta correctamente.
Requerimientos de ambiente de pruebas Hardware: circuito con los componentes IoT conectado al módulo de WIFI ESP32 Software: programación realizada en Arduino uno, en el servidor y en el aplicativo		**Condiciones / Restricciones** Ninguna.
Seguimiento		
Resultado obtenido Las operaciones CRUD de la fase se han realizado correctamente.	**Estado actual** Conforme	**Observaciones** Ninguna.
Correcciones		
Fecha de cambio de estado	**Observaciones**	

Ítem N°4

DESCRIPCIÓN DE PRUEBAS DE CAJA NEGRA		
Requisito		
Módulo / Área Funcional / Sub proceso Parámetros de temperatura, humedad ambiental y humedad de suelo	**Tipo de requisito** Funcional	**Código del requisito** -
Descripción del requisito El usuario debe poder gestionar los parámetros de temperatura, humedad ambiental y humedad de suelo de una planta mediante el aplicativo.		
Caso de prueba		
Código de prueba PCN03	**Caso de prueba** Gestión de parámetros	**Fecha de prueba** 10/05/2022
Funcionalidad / Característica a evaluar Gestionar parámetros	**Datos de entrada / Acciones de entrada** Operaciones CRUD de los parámetros	**Resultado esperado** El sistema listará, buscará, agregará, modificará y eliminará los parámetros de una planta correctamente.
Requerimientos de ambiente de pruebas Hardware: circuito con los componentes IoT conectado al módulo de WIFI ESP32 Software: programación realizada en Arduino uno, en el servidor y en el aplicativo		**Condiciones / Restricciones** Ninguna.
Seguimiento		
Resultado obtenido Las operaciones CRUD de los parámetros se han realizado correctamente.	**Estado actual** Conforme	**Observaciones** Ninguna.
Correcciones		
Fecha de cambio de estado	**Observaciones**	

Ítem N°4

DESCRIPCIÓN DE PRUEBAS DE CAJA NEGRA		
Requisito		
Módulo / Área Funcional / Subproceso Cultivo	**Tipo de requisito** Funcional	**Código del requisito** -
Descripción del requisito El usuario debe poder gestionar un cultivo mediante el aplicativo.		
Caso de prueba		
Código de prueba PCN03	**Caso de prueba** Gestión del cultivo	**Fecha de prueba** 10/05/2022
Funcionalidad / Característica a evaluar Gestionar cultivo	**Datos de entrada / Acciones de entrada** Operaciones CRUD del cultivo	**Resultado esperado** El sistema listará, buscará, agregará, modificará y eliminará un cultivo correctamente.
Requerimientos de ambiente de pruebas Hardware: circuito con los componentes IoT conectado al módulo de WIFI ESP32 Software: programación realizada en Arduino uno, en el servidor y en el aplicativo		**Condiciones / Restricciones** Ninguna.
Seguimiento		
Resultado obtenido Las operaciones CRUD del cultivo se han realizado correctamente.	**Estado actual** Conforme	**Observaciones** Ninguna.
Correcciones		
Fecha de cambio de estado	**Observaciones**	

Ítem N°5

DESCRIPCIÓN DE PRUEBAS DE CAJA NEGRA			
Requisito			
Módulo / Área Funcional / Subproceso Cosecha		**Tipo de requisito** Funcional	**Código del requisito** -
Descripción del requisito El usuario debe poder gestionar la cosecha de un cultivo mediante el aplicativo.			
Caso de prueba			
Código de prueba PCN03	**Caso de prueba** Gestión de cosecha		**Fecha de prueba** 10/05/2022
Funcionalidad / Característica a evaluar Gestionar cosecha	**Datos de entrada / Acciones de entrada** Operaciones CRUD de la cosecha	**Resultado esperado** El sistema listará, buscará, agregará, modificará y eliminará las cosechas de un cultivo correctamente.	
Requerimientos de ambiente de pruebas Hardware: circuito con los componentes IoT conectado al módulo de WIFI ESP32 Software: programación realizada en Arduino uno, en el servidor y en el aplicativo		**Condiciones / Restricciones** Ninguna.	
Seguimiento			
Resultado obtenido Las operaciones CRUD de la cosecha se han realizado correctamente.	**Estado actual** Conforme	**Observaciones** Ninguna.	
Correcciones			
Fecha de cambio de estado	**Observaciones**		

Ítem N°6

DESCRIPCIÓN DE PRUEBAS DE CAJA NEGRA		
Requisito		
Módulo / Área Funcional / Sub proceso Recordatorios	**Tipo de requisito** Funcional	**Código del requisito** -
Descripción del requisito El usuario debe poder gestionar el recordatorio de un cultivo mediante el aplicativo.		
Caso de prueba		
Código de prueba PCN03	**Caso de prueba** Gestión de Recordatorios	**Fecha de prueba** 10/05/2022
Funcionalidad / Característica a evaluar Gestionar recordatorio	**Datos de entrada / Acciones de entrada** Operaciones CRUD del recordatorio	**Resultado esperado** El sistema listará, buscará, agregará, modificará y eliminará los recordatorios de un cultivo correctamente.
Requerimientos de ambiente de pruebas Hardware: circuito con los componentes IoT conectado al módulo de WIFI ESP32 Software: programación realizada en Arduino uno, en el servidor y en el aplicativo		**Condiciones / Restricciones** Ninguna.
Seguimiento		
Resultado obtenido Las operaciones CRUD del recordatorio se han realizado correctamente.	**Estado actual** Conforme	**Observaciones** Ninguna.
Correcciones		
Fecha de cambio de estado	**Observaciones**	

Ítem N°7

ANEXO N° 08: Validación del Producto Acreditable

INSTRUMENTO DE VALIDACIÓN

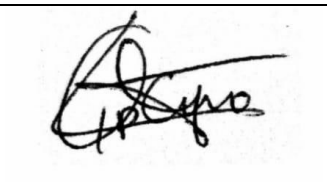
Documento para medir la funcionalidad	
Elaborado por:	Paola Patricia Castro Fernández
Revisado por:	Ing. Chavarry Chankay Mariana
Aprobado por:	Ing. Chavarry Chankay Mariana
Objetivo de la Ficha	
Calificar la funcionalidad del Sistema IoT desarrollado.	
Procedimiento	
El experto debe evaluar la funcionalidad del Sistema IoT.	
Nombre del Experto:	Ing. Guadalupe Lip Curo
Indicación:	Calificación de la funcionalidad del Sistema IoT
Criterios de evaluación:	Si: Se marca cuando el requerimiento indicado se este cumpliendo en un 100% No: Se marca cuando el requerimiento indicado no se está cumpliendo en su totalidad.

Para el módulo IoT			
N°	**Pregunta**	**Criterio**	
		Si	**No**
1	¿Se validó la obtención de los datos del sensor de temperatura y humedad?	X	
2	¿Se validó la obtención de los datos de los sensores de humedad de suelos?	X	
3	¿Se validó la obtención de los datos del sensor de lluvia?	X	
4	¿Se validó la obtención de los datos del sensor de luminosidad?	X	
5	¿Se validó el accionar de los actuadores tomando en cuenta los parámetros establecidos?	X	
6	¿Se validó la transmisión de los datos de los sensores y actuadores al aplicativo móvil?	X	
Para el módulo de Inicio			
N°	**Pregunta**	**Criterio**	
		Si	**No**
7	¿Se válido la visualización de los datos de los sensores en tiempo real o el último	X	

	registro de los datos (en caso de estar el módulo IoT desconectado)?		
Para el módulo de Plantas			
N°	**Pregunta**	**Criterio**	
		Si	**No**
8	¿Se registra correctamente los datos de la planta?	X	
9	¿Se registra correctamente los datos de las fases de la planta?	X	
10	¿Se registra correctamente los datos de los parámetros de la planta?	X	
11	¿Se listan correctamente las plantas registradas?	X	
12	¿Se listan correctamente las fases registradas de una planta en específico?	X	
13	¿Se listan correctamente los parámetros registrados de una planta en específico?	X	
14	¿Se modifican correctamente los datos de una determinada planta registrada?	X	
15	¿Se modifican correctamente los datos de una determinada fase registrada de una planta en específico?	X	
16	¿Se modifican correctamente los datos de un determinado parámetro registrado de una planta en específico?	X	
17	¿Se eliminan correctamente las plantas registradas si estas no tienen parámetros, fases o cultivos registrados?	X	
18	¿Se eliminan correctamente las fases registradas siempre y cuando no esté relacionado a un cultivo?	X	
19	¿Se eliminan correctamente los parámetros registrados de una planta determinada?	X	
Para el módulo de Cultivos			
N°	**Pregunta**	**Criterio**	
		Si	**No**

20	¿Se registra correctamente los datos de cultivo?	X	
21	¿Se listan correctamente los cultivos registrados?	X	
22	¿Se modifican correctamente los cultivos registrados?	X	
23	¿Se eliminan correctamente los cultivos registrados en caso no tengan cosechas registradas?	X	
Para el módulo de Cosechas			
N°	**Pregunta**	**Criterio**	
		Si	**No**
24	¿Se registra correctamente los datos de la cosecha?	X	
25	¿Se listan correctamente las cosechas registradas?	X	
26	¿Se modifican correctamente los datos de las cosechas registradas?	X	
27	¿Se eliminan correctamente las cosechas?	X	
Para el módulo de Recordatorios			
N°	**Pregunta**	**Criterio**	
		Si	**No**
28	¿Se registran correctamente los datos del recordatorio?	X	
29	¿Se listan correctamente los recordatorios registrados?	X	
30	¿Se modifican correctamente los datos de los recordatorios registrados?	X	
31	¿Se eliminan correctamente los recordatorios?	X	
Para el módulo de Reportes			
N°	**Pregunta**	**Criterio**	
		Si	**No**
32	¿Se pueden filtrar los reportes por año y mes?	X	
33	¿Se visualiza correctamente los gráficos?	X	
Para el módulo de Actuadores			
N°	**Pregunta**	**Criterio**	
		Si	**No**
34	¿Se puede visualizar el estado actual de los	X	

	actuadores o el último registro de los datos (en caso de estar el módulo IoT desconectado)?		

Observaciones	
El reporte puede ser mejorado poniendo indicadores específicos que aporten a la toma de decisiones.	
	
Firma del Experto	**Firma del Tesista**

INSTRUMENTO DE VALIDACIÓN

Documento para medir la funcionalidad	
Elaborado por:	Paola Patricia Castro Fernández
Revisado por:	Ing. Chavarry Chankay Mariana
Aprobado por:	Ing. Chavarry Chankay Mariana
Objetivo de la Ficha	
Calificar la funcionalidad del Sistema IoT en el cultivo de tomates nativos.	
Procedimiento	
El experto debe evaluar la funcionalidad del Sistema IoT en el cultivo de tomates nativos.	
Nombre del cultivador/Ingeniero agrónomo:	Ing. Fernando Alvarado Reyna
Indicación:	Calificación de la funcionalidad del Sistema IoT en el cultivo de tomates nativos
Criterios de evaluación:	Si: Se marca cuando el requerimiento indicado se este cumpliendo en un 100% No: Se marca cuando el requerimiento indicado no se está cumpliendo en su totalidad.

Para el módulo IoT			
N°	**Pregunta**	**Criterio**	
		Si	**No**
1	¿Se validó la obtención de los datos de los sensores del circuito conectado al ambiente del cultivo cumpliendo con los parámetros establecidos?	X	
2	¿Se validó el accionar de los actuadores con respecto a los parámetros establecidos para el cultivo?	X	
Para el módulo de Inicio			
N°	**Pregunta**	**Criterio**	
		Si	**No**
3	¿Se válido la visualización de los datos de los sensores en tiempo real o el último registro de los datos (en caso de estar el módulo IoT desconectado)?	X	
Para el módulo de Plantas			
N°	**Pregunta**	**Criterio**	
		Si	**No**
4	¿Los datos registrados de la planta son los adecuados?	X	
5	¿Los datos registrados de las fases son los adecuados?	X	

6	¿Los datos registrados por los parámetros son los adecuados?	X	
7	¿Se listan correctamente las plantas registradas?	X	
8	¿Se listan correctamente las fases registradas de una determinada planta?	X	
9	¿Se listan correctamente los parámetros registrados de una determinada planta?	X	
10	¿Se modifican correctamente los datos de las plantas registradas?	X	
11	¿Se modifican correctamente los datos de las fases registradas de una determinada planta?	X	
12	¿Se modifican correctamente los datos de los parámetros registrados de una determinada planta?	X	
13	¿Se eliminan correctamente las plantas registradas si estas no tienen parámetros, fases o cultivos registrados?	X	
14	¿Se eliminan correctamente las fases registradas siempre y cuando no esté pasando un cultivo por esa fase?	X	
15	¿Se eliminan correctamente los parámetros registrados?	X	
Para el módulo de Cultivos			
N°	**Pregunta**	**Criterio**	
		Si	**No**
16	¿Se registra los datos del cultivo adecuados?	X	
17	¿Se listan correctamente los cultivos registrados?	X	
18	¿Se modifican correctamente los cultivos registrados?	X	
19	¿Se eliminan correctamente los cultivos registrados en caso no tengan cosechas registradas?	X	
Para el módulo de Cosechas			
N°	**Pregunta**	**Criterio**	
		Si	**No**

20	¿Se registran los datos de la cosecha adecuados?	X	
21	¿Se listan correctamente las cosechas registradas?	X	
22	¿Se modifican correctamente las cosechas registradas?	X	
23	¿Se eliminan correctamente las cosechas?	X	
Para el módulo de Recordatorios			
N°	**Pregunta**	**Criterio**	
		Si	**No**
24	¿Se registra los datos adecuados para los recordatorios?	X	
25	¿Se listan correctamente los recordatorios registrados?	X	
26	¿Se modifican correctamente los recordatorios registrados?	X	
27	¿Se eliminan correctamente los recordatorios?	X	
Para el módulo de Reportes			
N°	**Pregunta**	**Criterio**	
		Si	**No**
28	¿Se pueden filtrar los reportes por año y mes?	X	
29	¿Son útiles los gráficos presentados?	X	
Para el módulo de Actuadores			
N°	**Pregunta**	**Criterio**	
		Si	**No**
30	¿Se puede visualizar el estado actual de los actuadores o el último registro de los datos (en caso de estar el módulo IoT desconectado)?	X	
Para el Cultivo de tomates			
N°	**Pregunta**	**Criterio**	
		Si	**No**
31	¿Se validó el correcto estado del cultivo de acuerdo con las fases de la planta?	X	
32	¿Se validó que el cultivo no conto ni cuenta con plagas?	X	

33	¿Se validó que el cultivo no conto ni cuenta con enfermedades?	X	

Observaciones	
Cuando hay épocas de lluvia, se ha considerado; que la aplicación detecte: * Número de plantaciones afectadas o perdidas en época de lluvia. * Número de plantaciones que se contaminaron con plagas. Muestra es información reportando esa información de los diferentes estados.	
	
Firma del Evaluador	**Firma del Tesista**