UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic

Trabajo de Titulación para optar al título de Ingeniero en tecnologías de la información

Autor:
Manrique Arias Lennin Andres

Tutor:
Ing. Pamela Alexandra Buñay Guisñan

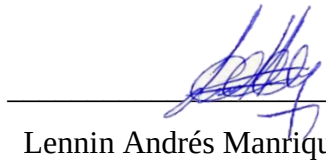
Riobamba, Ecuador. ***2024***

DECLARATORIA DE AUTORÍA

Yo, Lennin Andres Manrique Arias, con cédula de ciudadanía 1600463754, autor del trabajo de investigación titulado: Prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor de la obra referida será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, a los 10 días del mes de mayo del 2024.



Lennin Andrés Manrique Arias

C.I: 1600463754

DICTAMEN FAVORABLE DEL PROFESOR TUTOR

Dirección
Académica
VICERRECTORADO ACADÉMICO

en movimiento
SGC
SISTEMA DE GESTIÓN DE LA CALIDAD

ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN

En la Ciudad de Riobamba, a los 26 días del mes de abril de 2024, luego de haber revisado el Informe Final del Trabajo de Investigación presentado por el estudiante **Lennin Andrés Manrique Arias** con CC: **160046375-4**, de la carrera **de Ingeniería en Tecnologías de la Información** y dando cumplimiento a los criterios metodológicos exigidos, se emite el **ACTA FAVORABLE DEL INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN** titulado **"Prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic"**, por lo tanto se autoriza la presentación del mismo para los trámites pertinentes.

Firmado electrónicamente por:
PAMELA ALEXANDRA
BUNAY GUISNAN

Ing. Pamela Buñay
TUTOR(A)

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación "Prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic", presentado por Lennin Andrés Manrique Arias con cédula de identidad número 160046375-4, bajo la tutoría de Ing. Pamela Alexandra Buñay; certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba a los 30 días del mes de mayo del 2024

Ing. Lady Espinoza, Mgs
PRESIDENTE DEL TRIBUNAL DE GRADO

Ing. Estela Narváez, PhD
MIEMBRO DEL TRIBUNAL DE GRADO

Ing. Ximena Quintana, PhD
MIEMBRO DEL TRIBUNAL DE GRADO

CERTIFICADO ANTIPLAGIO

Dirección
Académica
VICERRECTORADO ACADÉMICO

en movimiento
SGC
SISTEMA DE GESTIÓN DE LA CALIDAD
UNACH-RGF-01-04-08.17
VERSIÓN 01: 06-09-2021

C E R T I F I C A C I Ó N

Que, **MANRIQUE ARIAS LENNIN ANDRES** con CC: **1600463754**, estudiante de la Carrera **INGENIERIA EN TECNOLOGIAS DE LA INFORMACIÓN**, Facultad de **INGENIERA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado "**Prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic**", cumple con el 10 %, de acuerdo al reporte del sistema Anti plagio **TURNITIN**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 15 de mayo de 2024

Firmado electrónicamente por:
PAMELA ALEXANDRA
BUNAY GUISNAN

Mgs. PAMELA ALEXANDRA BUÑAY GUISÑAN
TUTOR

DEDICATORIA

Quiero dedicar esto a mi familia en general, ya que han sido de sumo apoyo para el desarrollo de este proyecto final y a lo largo de toda la carrera universitaria. Me encuentro agradecido por brindarme el mejor regalo que un hijo puede tener de sus padres que es la educación, y por el incondicional apoyo en el transcurso de estos años.

A mis profesores quienes gracias a su orientación han sido fundamentales para el desarrollo de este proyecto de investigación.
A mis amigos, quienes me han apoyado en mis momentos difíciles siendo una segunda familia, consejeros y en ocasiones pilares para no rendirse en este largo camino que se llama vida.
A “” mi mejor amigo y compañero de clase que ha sido un ejemplo de superación en momentos difíciles y situaciones que parecían no tener una solución positiva.

Lennin Andrés Manrique Arias

“No importa lo lento que vayas, siempre y cuando no te detengas.”
-Confucio

AGRADECIMIENTO

Quisiera expresar mi profunda gratitud a mi madre y padre por su apoyo incondicional y por no dudar de mi en ningún momento. Gracias por su amor y comprensión a lo largo de todos estos años.

De igual manera quiero agradecer a la Universidad Nacional de Chimborazo y a la carrera de Tecnologías de la información, por brindarme la oportunidad de forjar mis conocimientos profesionales dentro de sus laboratorios y aulas de clase. Agradezco también a mis profesores que no dudaron al momento de compartir sus conocimientos en clase.

Quiero agradecer a mi amigo Isaí Leopoldo graduado de la carrera de Electrónica y Telecomunicaciones por su colaboración en el desarrollo del prototipo final de mi proyecto de investigación.

Agradezco también a la Cooperativa de Producción Agrícola Ananda por brindarme un espacio dentro de sus oficinas y equipo electrónico para el desarrollo de mi proyecto de investigación.

Finalmente quiero agradecer a todas las personas sean amistades, parejas o colegas que me han motivado cada día a seguir adelante ya sea aconsejando o apoyando de cualquier manera que les ha sido posible.

Lennin Andrés Manrique Arias

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto de investigación tuvo como objetivo principal desarrollar un prototipo de internet de las cosas (IoT) para el monitoreo del cultivo de tomate en un huerto urbano mediante el desarrollo de una aplicación móvil usando el framework Ionic. Permitiendo que el usuario pueda tomar decisiones sobre el cultivo en base a los datos recolectados por el dispositivo. Para desarrollar la aplicación móvil se utilizó el framework Ionic y una base de datos en MongoDB Atlas, Express.js como el framework para construir la parte del servidor de la aplicación móvil.

Este trabajo de investigación se enfocó en un análisis cuantitativo, porque se obtuvieron datos medibles sobre el rendimiento de la aplicación móvil que fue desarrollada a través del modelo de calidad FURPS. La metodología desarrollo que se utilizó para el proyecto fue Waterfall o cascada otorgando una estructura secuencial que permitió un enfoque sistemático en el desarrollo, asegurando una planificación clara, lo que fue importante para cumplir con los objetivos establecidos. Se evaluó la aplicación móvil mediante el modelo de calidad FURPS usando el enfoque de fiabilidad el cual en sus resultados muestra que hay un 100% de recuperabilidad, un 87% de predicción y 100% en el criterio de precisión.

Palabras claves: Aplicación móvil, cascada, cultivo urbano, ESP32, Ionic Framework, IoT, MongoDB

ABSTRACT

ABSTRACT

The main objective of this research project was to develop an Internet of Things (IoT) prototype for monitoring the tomato crop in an urban garden by creating a mobile application using the Ionic framework. The application would allow the user to make decisions about the crop based on the data collected by the device. To develop the mobile application, we used the Ionic framework and a database in MongoDB Atlas, with Express.js as the framework to build the server side of the mobile application.

The development methodology employed for this project was Waterfall, which provided a sequential structure that allowed for a systematic approach to development. This approach ensured precise planning, which was crucial in meeting the established objectives. The mobile application, a key component of our IoT prototype, was rigorously evaluated using the FURPS quality model. The results were highly encouraging, demonstrating 100% recoverability, 87% predictability, and a perfect 100% in the accuracy criterion, affirming the effectiveness of our approach.

Keywords: mobile app, Waterfall, urban farming, ESP32, Ionic Framework, IoT, MongoDB

Reviewed by:
Ms.C. Ana Maldonado León
ENGLISH PROFESSOR
C.I.0601975980

CAPÍTULO I. INTRODUCCION

El desarrollo de Internet y su uso es uno de los factores que más está afectando en el funcionamiento de la sociedad. Es enorme el impacto social que están teniendo las nuevas tecnologías y las transformaciones que estas provocan en ámbitos tan comunes como es el trabajo, industria o consumo. A medida que surgen avances en tecnología y el incremento del consumo de dispositivos inteligentes se confirma la tendencia hacia el "Internet de las Cosas (IoT)". Esta nueva tendencia está centrada en las personas, procesos y objetos. El Internet de las Cosas (IoT) hace referencia a una red de objetos de uso cotidiano, interconectados entre sí. Se trata de una nueva tendencia que surge tras el desarrollo del Smartphone, y que afecta a diferentes sectores, como la industria, la sanidad, la administración pública o la agricultura. (Consulting Informático, 2021)

Hoy en día, alrededor de la mitad de la población mundial vive en ciudades y se cree que esta tendencia continuará. Según pronósticos la población urbana se duplicará en los próximos treinta años dando como resultado que siete de cada diez personas vivan en ciudades (Banco Mundial, 2022)

Los desarrollos en la ciencia y tecnología han permitido en los últimos tiempos, mejorar las condiciones de vida de los seres humanos. Esto ha permitido que se puedan mejorar muchos de los procesos en el vivir diario de la gente, incluida la agricultura y todo tipo de cultivos. Dada la importancia que tiene la agricultura dentro de la supervivencia, desde hace tiempo se trabaja en sistemas que permitan optimizar procesos, para lograr una mejora en la producción. Estos procesos han estado estrechamente vinculados con mejoras genéticas de algunas plantas, eliminación de plagas, técnicas de cultivo entre otros. En muchas regiones del mundo los cambios extremos en el clima afectan directamente a la productividad de los cultivos. A raíz de estas situaciones, con el tiempo se han desarrollado nuevas tecnologías que permiten conocer el estado de los cultivos, que pueden ir desde la temperatura, condiciones de humedad del suelo y niveles de radiación. La idea de implementar tecnologías relacionadas con el internet de las cosas (IoT), la recolección de datos, el monitoreo y evaluación de un sistema de cultivo resulta ser imprescindible para tomar decisiones acertadas. (Banco Mundial, 2022)

Medir estas variables resulta ser de mucha utilidad para un adecuado control de los cultivos. Por ejemplo, si se mide un bajo nivel en la tierra se activan los aspersores de agua hasta el nivel requerido para un cultivo especifico. Para controlar las variables es necesario tener un espacio cerrado como un invernadero. Así, es posible tener cultivos productivos independientemente de las condiciones ambientales de la zona geográfica donde se encuentre. (IAT, 2020)

Actualmente, Riobamba no cuenta con sistemas de monitoreo que permitan a las personas seguir el progreso de sus prácticas hortícolas. Por este motivo se ha desarrollado un prototipo Iot de monitoreo que permita conocer el estado de los huertos urbanos mediante una aplicación móvil que se encargue de analizar los datos recopilados y se pueda tomar decisiones. La aplicación móvil será desarrollada usando la metodología de desarrollo en cascada y finalmente se evaluará el rendimiento de la aplicación. (IAT, 2020)

PLANTEAMIENTO DEL PROBLEMA

1.1 Problema

Debido al crecimiento de la población urbana, cada año el tamaño de las ciudades aumenta, generando una necesidad de recursos alimenticios, plantea un problema de acceso, disponibilidad de alimentos y una buena nutrición, especialmente para las personas con bajos ingresos. Como solución a nivel global surgen los cultivos urbanos, que permite a una comunidad tener una fuente de auto sostenimiento e ingresos adicionales. El mantenimiento de estos cultivos requiere de tiempo y recursos por parte del productor. (Degenhart, 2016)

Los huertos urbanos existen desde hace millones de años, durante el Neolítico las mujeres cultivaban semillas alrededor de su casa, aunque en la Segunda Guerra Mundial comenzaron a desarrollarse los huertos urbanos para asegurarse los alimentos. En Estados Unidos, Reino Unido y Alemania se llegaron a utilizar los campos de fútbol o los parques para cultivar. (educo, 2019)

Para mantener protegidos a los cultivos en el interior de las viviendas hay que tomar precauciones. El frio afecta a las plantas y a sus contenedores que dependiendo del clima extremo se pueden expandir o agrietar. Cuando hace un calor excesivo el suelo se reseca y pierde humedad, las plantas tienden a resecarse haciendo que sus hojas se deterioren rápidamente y caigan. En ciertas plantas pueden recuperarse en pocas semanas, pero otras mueren. (hidroenv, 2018)

La luz desempeña un papel importante en el desarrollo de una planta. Procesos como fotosíntesis y fototropismo dependen de la disponibilidad de las fuentes de luz, son necesarios para que el tomate pueda tener energía, crecer y sobrevivir. Tecnificar y hacer más eficiente el seguimiento de los cultivos, reducirá los costos de producción y mejorará la calidad final del producto, generando un beneficio a la comunidad dentro de la ciudad de Riobamba.
Por dichas problemáticas y enfatizando que la gente no tiene suficiente tiempo de ocuparse de sus cultivos caseros, se ha considerado el uso de las tecnologías de la información mediante la incorporación de aplicaciones móviles, además del uso de sensores de IoT, para poder monitorear de manera rápida la información referente al crecimiento de las plantas de tomate en un huerto urbano. Para que mediante la información que se obtiene al monitorear los parámetros del cultivo se pueda optimizar el control de este minimizando considerablemente el tiempo en el que la planta hortícola se desarrolla y da sus frutos.

1.2 Formulación del problema

¿Cómo la implementación de un prototipo de IoT con almacenamiento en la nube y soporte en una aplicación móvil usando Ionic es fiable en la agricultura inteligente?

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic.

1.3.2 Objetivos específicos

- Investigar las aplicaciones de IoT para el monitoreo de cultivos.
- Implementar el prototipo IoT usando una aplicación móvil con Ionic para el monitoreo del cultivo de tomate.
- Evaluar la fiabilidad de la aplicación móvil para el monitoreo del cultivo de tomate usando el modelo de calidad FURPS.

CAPÍTULO II. MARCO TEÓRICO

2.1 Internet de las cosas

Es el proceso que permite conectar los elementos físicos cotidianos a internet desde los objetos domésticos comunes, como las bombillas de luz, hasta los recursos para la atención de la salud, como los dispositivos médicos; las prendas y los accesorios personales inteligentes; e incluso los sistemas de las ciudades inteligentes. (RedHat, 2019)
Esta tecnología permite obtener información basada en datos, favorece en la productividad, eficiencia y a su vez en la prevención de fallas técnicas. El ámbito de aplicación de la IoT es bastante amplio, abarca desde la industria, fabricación, salud, energía, agricultura, entre otros. En cada uno de estos sectores ha significado un aliciente para seguir desarrollando interesantes dispositivos que facilitan la vida. (RedHat, 2019)

2.1.1 Tecnologías que lo han hecho posible

Si bien la idea de IoT existe desde hace mucho tiempo, una colección de avances recientes en una serie de tecnologías diferentes la ha hecho práctica. (Oracle, 2022)

Accede a la tecnología de sensores de bajo costo y potencia

Los sensores asequibles y fiables están haciendo posible la tecnología IoT para más fabricantes. (Oracle, 2022)

Conectividad

Una gran cantidad de protocolos de red para Internet ha facilitado la conexión de sensores a la nube y a otras cosas para lograr una transferencia de datos eficiente (Oracle, 2022)

Plataformas de informática en la nube

El aumento en la disponibilidad de plataformas en la nube permite a las empresas y a los consumidores acceder a la infraestructura que necesitan para escalar sin tener que administrarlo todo. (Oracle, 2022)

Aprendizaje automático y analítica

Con los avances en aprendizaje automático y analítica, junto con el acceso a cantidades grandes y variadas de datos almacenados en la nube, las empresas pueden recopilar información de forma más rápida y fácil. Por un lado, el surgimiento de estas tecnologías aliadas sigue traspasando los límites de IoT; por otro, los datos producidos por IoT también alimentan estas tecnologías. (Oracle, 2022)

Inteligencia artificial (IA) conversacional

Los avances en las redes neuronales han llevado el procesamiento del lenguaje natural (PLN) a los dispositivos IoT (como los asistentes personales digitales Alexa, Cortana y Siri) y los han hecho atractivos, asequibles y viables para uso doméstico. (Oracle, 2022)

2.1.2 Esquema del Internet de las cosas (IoT)

En la imagen 1 que se presenta a continuación se muestra un diagrama básico de IoT. En este diagrama, los dispositivos IoT se conectan al servidor IoT a través de Internet. El servidor IoT actúa como un punto centralizado para recibir, procesar y almacenar datos de los dispositivos IoT. Los dispositivos IoT pueden enviar datos al servidor a través de protocolos de comunicación estándar, como MQTT o HTTP, y el servidor puede responder

con comandos o actualizaciones para los dispositivos. Los datos recopilados por el servidor IoT también se pueden analizar y visualizar a través de una interfaz de usuario o una API, lo que permite a los usuarios monitorear y controlar sus dispositivos IoT desde cualquier lugar con acceso a Internet.

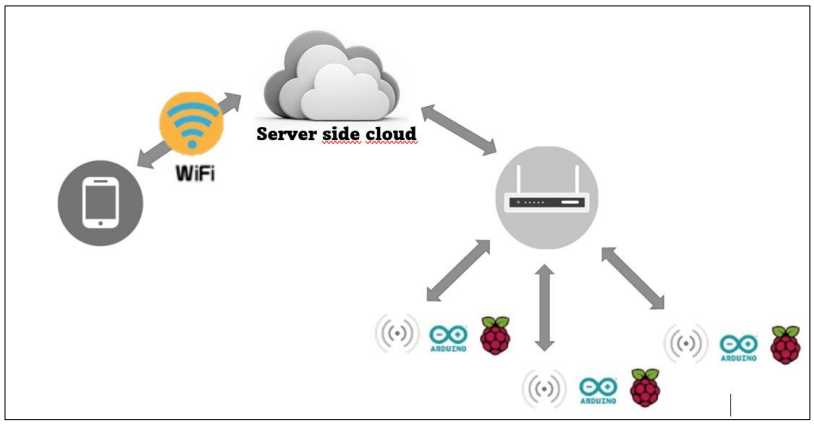


Figura 1: Diagrama de un sistema IoT básico (Nuñez, 2017)

2.2 IoT en la agricultura

Agricultura inteligente

Puede definirse como uso y empleo tecnológico para cultivar alimentos de manera limpia y sostenible.

En la figura 2 se representa una idea básica de la agricultura inteligente, en esta imagen una estación base se conecta a múltiples sensores y actuadores que monitorean y controlan el entorno de cultivo, como la humedad del suelo, la temperatura, la humedad del aire, la luz y los niveles de CO2. Los datos recopilados por los sensores se envían al sistema de control, que puede ajustar los actuadores para mantener las condiciones de crecimiento óptimas para los cultivos. El sistema de control también puede enviar alertas a los usuarios si se detectan problemas en el cultivo. Cada cultivo está equipado con sus propios sensores y actuadores conectados al sistema de control, lo que permite el monitoreo y control individualizado de cada uno. Además, se puede utilizar una cámara de video para monitorear visualmente el crecimiento de los cultivos.

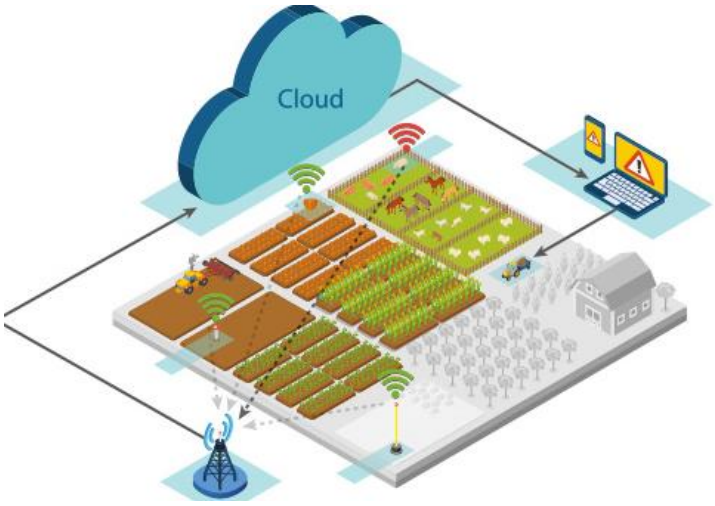


Figura 2: Diagrama de la agricultura inteligente

Fuente: (García, 2022)

En definitiva, consiste en la incorporación de procesos tecnológicos para beneficiar la productividad de los campos agrícolas y obtener una producción de forma sustentable. (App&Web, 2019)

Tabla 1: Soluciones de IoT en la agricultura

Soluciones de IoT en la agricultura	
Invernaderos inteligentes	Los invernaderos tradicionales tienen la función de crear determinados ambientes climatológicos para favorecer el desarrollo de determinados alimentos y plantaciones.
Drones agrícolas	Los drones son dispositivos que cada vez son más usado en el terreno de la agricultura. Estas aeronaves pueden volar sobre los campos de cultivo y recoger una gran cantidad de información sobre los mismos.
Monitorización y precisión	Mediante la aplicación de IoT en el sector de la agricultura también se pueden incorporar sensores que registren datos y permitan medir y evaluar la evolución de la producción.
Control inteligente de plagas	Las plagas o enfermedades que pueden atacar a los campos de cultivo pueden suponer un gran riesgo para los agricultores, ya que pueden deteriorar o arrollar con la totalidad de la producción.
Gestión eficiente de recursos	Dado que el IoT o Internet de las Cosas permite tener un conocimiento total sobre las condiciones del cultivo, podremos hacer un uso mucho más eficiente de los recursos, especialmente del agua.

Fuente: (App&Web, 2019)

2.3 Huerto urbano

Es un espacio ubicado generalmente en el exterior y destinado a cultivar verduras, hortalizas, legumbres, plantas aromáticas a pequeña escala y, generalmente, destinadas al autoconsumo, así como se muestra en la figura 3 (educo, 2019)

Figura 3: Huerto urbano

Beneficios de un huerto urbano

- Un huerto urbano proporciona alimentos sanos y nutritivos, con el sabor auténtico de las hortalizas.
- Es una actividad relajante y saludable, que implica hacer un poco de ejercicio aeróbico, lo cual tiene consecuencias muy favorables para la salud.
- Es una manera de valorar el medio ambiente y la naturaleza, poniéndonos en una mayor conexión con ella.
- Es una actividad muy adecuada para todo el mundo, incluidos niños y mayores. Los primeros encuentran en ella una gran fuente de conocimiento, mientras que para las personas mayores es una buena manera de hacer ejercicio.
- La jardinería urbana sensibiliza ante los procesos productivos y la calidad de las frutas y hortalizas que compramos en el mercado o el súper, teniendo aún más en cuenta no solo su sabor y calidad, sino también sus métodos de producción, valorando más los cultivos ecológicos.
- Sirven para conocer mejor los procesos naturales y potencian nuestra capacidad para observar el medio natural
- En las ciudades y zonas urbanas, se convierten en pequeños pulmones, que purifican el aire.

Tabla 2: Tipos de huertos urbanos

Tipos de huertos	
Huerto urbano vertical	son la mejor opción para espacios exteriores pequeños, en los que no se puede instalar una mesa de cultivo o un huerto de suelo. Van instalados en las paredes mediante módulos de polipropileno o de yute y son muy resistentes a la intemperie.
Mesa de cultivo	están diseñados con profundidad suficiente para acoger un gran volumen de sustrato. Su éxito reside en su comodidad, ya que evitan

	que haya que agacharse para trabajar en él. Suelen emplearse para hortalizas que no crecen mucho en altura, como lechugas, pimientos o berenjenas. La mayoría incluyen un estante inferior, lo que ayuda a tener todas las herramientas debajo guardadas y, lo más importante, a mano. Son una buena solución para iniciados en los huertos urbanos, además de
Huerto de suelo	son grandes contenedores que se instalan en el suelo. Pueden ser planos o contar con escalones, ofreciendo un atractivo resultado cuando el huerto está en pleno esplendor. Debido a su gran tamaño, suelen ser las más adecuadas para jardines o áticos, en los que los metros cuadrados no son un problema.

Fuente: (Lavín, 2022)

2.4 Sensores IoT

Tipos

Los sensores se pueden clasificar en función de varias características, pero el criterio de selección más importante es la magnitud, estas son las métricas más comunes. (Tokioschool, 2022)

- Temperatura
- Humedad
- Presión
- Niveles de depósitos/acumuladores
- Caudales de línea o servicios
- Presencia/contaje de piezas/paquetes/bolsas
- Tensión, consumo y potencia eléctrica
- Proximidad
- Luminosidad
- Ruido

Aplicaciones

Una cuestión importante para la captura de los datos de forma correcta es la instalación y conexión de los sensores a nivel de campo y cómo se hace para evitar inversiones innecesarias en sensores replicados. En todas las fábricas existen sensores conectados a sistemas SCADA/PLC que hacen uso de la información para sus operaciones rutinarias. Estos sistemas se comportan como "silos de información" debido a que los datos recogidos están disponibles para la toma de decisiones por parte de la máquina o para monitorizar de forma local en monitores o salas de control y supervisión (a nivel de campo). (Tokioschool, 2022)

Diagrama de sensores IoT

Los sensores IoT pueden recopilar una amplia variedad de tipos de datos, dependiendo de su función y aplicación específicas. Algunos ejemplos comunes de tipos de datos que los sensores IoT pueden recopilar se muestran en la figura 4.

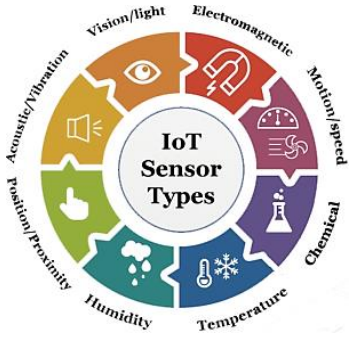


Figura 4: Tipos de datos que pueden recoger los sensores IoT
Fuente: (Lanner, 2016)

2.5 Arduino IDE

IDE (sigla en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. (Carmenate, 2021)

Figura 5: Interfaz principal del IDE de Arduino con placa de desarrollo

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Además, en el caso de Arduino incorpora las herramientas para cargar el programa ya compilado en la memoria flash del hardware, La interfaz principal del entorno de programación de Arduino se muestra en la figura 5. (Carmenate, 2021)

Los programas de Arduino están compuestos por un solo fichero con extensión “ino”, aunque es posible organizarlo en varios ficheros. El fichero principal siempre debe estar en una carpeta con el mismo nombre que el fichero.

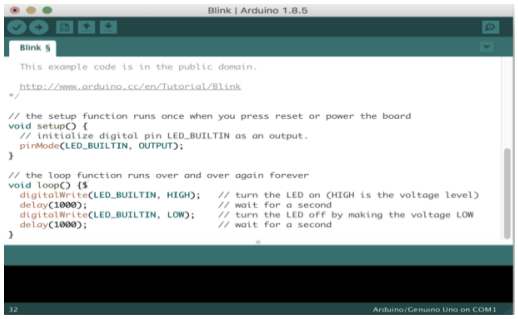


Figura 6: Estructura básica del código Arduino
Fuente: (flexbot, 2020)

En la figura 6 se muestra un diseño básico de código en el entorno de desarrollo de Arduino, el código mostrado se encarga de encender y apagar con un retraso de un segundo el led que viene incluido en la placa Arduino.

2.6 Plataforma IoT

Se entiende como plataforma IoT a un conjunto de tecnologías, servicios y herramientas que facilitan el desarrollo, la implementación y la gestión de aplicaciones y dispositivos conectados en el contexto de la internet de las cosas, permitiéndoles recopilas y compartir datos. Algunas de las funciones clave de una plataforma IoT incluyen:

- **Conectividad**

Proporciona capacidades de conexión entre dispositivos y sensores.

- **Gestión de dispositivos**

Permite incorporar, configurar y gestionar de manera eficiente dispositivos IoT.

- **Almacenamiento y gestión de datos**

Gestiona los grandes volúmenes de datos generados por los dispositivos.

- **Seguridad**

Ofrece autenticación, autorización y cifrado para las comunicaciones.

- **Procesamiento y análisis de datos**

Capacidad para realizar procesamiento de datos en tiempo real. (cognizant, 2023)

- **Interfaz de usuario y experiencia del desarrollador**

Interfaz tanto para usuarios finales como para desarrolladores.

- **Escalabilidad**

Ofrece flexibilidad para adaptarse a cambios en la infraestructura.

- **Integración con otros sistemas**

Se integra con sistemas existentes, como bases de datos, sistemas de gestión empresarial (ERP) y sistemas de información geográfica (GIS).

- **Desarrollo de aplicaciones**

Mediante la provisión de Apis y kits de desarrollo (SDK).

2.7 Protocolos de comunicación IoT

En primer lugar, en el IoT se puede llegar a tener una gran cantidad de dispositivos. Algunos de ellos serán pequeños (de pocos recursos), como sensores o actuadores. Mientras que otros serán más grandes, como un servidor que recoge información, almacena datos, y procesa estadísticas. (barbaraiot, 2021)

Los protocolos IoT son un conjunto de normas y reglas que permiten a dos entidades entenderse e intercambiar información facilitando la comunicación Machine2Machine (M2M). En otras palabras, los protocolos IoT son para la comunicación entre máquinas lo que los idiomas, los gestos o el lenguaje corporal son para la comunicación entre humanos.
Los principales protocolos IoT más conocidos en la tabla 3.

Tabla 3: Protocolos de comunicación IoT más comunes

Protocolos de IoT
MQTT. Un protocolo de mensajería diseñado para una comunicación ligera M2M y que se utiliza principalmente para conexiones de ancho de banda bajo en ubicaciones remotas. MQTT utiliza un patrón de publicador-suscriptor y es ideal para dispositivos pequeños que requieren un uso eficiente de la batería y el ancho de banda. (Bassi, 2021)
AMQP. Una capa de software que crea interoperabilidad entre middleware de mensajería. Ayuda a que una variedad de sistemas y aplicaciones trabajen juntos, creando mensajes estandarizados a escala industrial. (Bassi, 2021)
CoAP. Es un protocolo diseñado para entornos con un ancho de banda restringido y dispositivos con capacidad limitada. CoAP es también un protocolo de transferencia de documentos que se ejecuta sobre el UDP. (Bassi, 2021)

Fuente: (azure microsoft, 2023)

2.8 MongoDb Atlas

Ofrece a los clientes un servicio completamente administrado basado en la infraestructura confiable y escalable a nivel global de Google. Atlas permite administrar bases de datos con facilidad mediante unos pocos clics en la IU o a través de una llamada a la API, se puede migrar sin demasiado trabajo y ofrece funciones avanzadas como clústeres globales para obtener acceso a operaciones de lectura y escritura de baja latencia desde cualquier parte del mundo. (google Cloud, 2020)

En la figura 8 se muestra un diagrama básico del uso de MongoDB Atlas en una arquitectura IoT

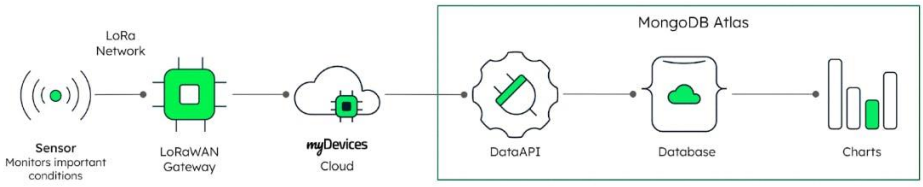


Figura 7: Diagrama de MongoDB Atlas en IoT

2.9 Plataforma Firebase

Es una plataforma digital diseñada para facilitar el desarrollo de aplicaciones web y móviles de calidad de una forma rápida y eficiente, con el objetivo de mejorar el rendimiento de estas a través de la implementación de sus distintos módulos que harán que la aplicación sea mucho más manejable, segura y fácil de utilizar para los usuarios.

Figura 8: Herramientas que ofrece la plataforma firebase

Fuente: (Andrade, 2022)

Multiplataforma

Firebase ofrece documentación detallada y SDK multiplataforma que te ayudarán a compilar y lanzar apps en Android, iOS, la Web, C++ y Unity como se muestra en la figura 9 (Andrade, 2022)

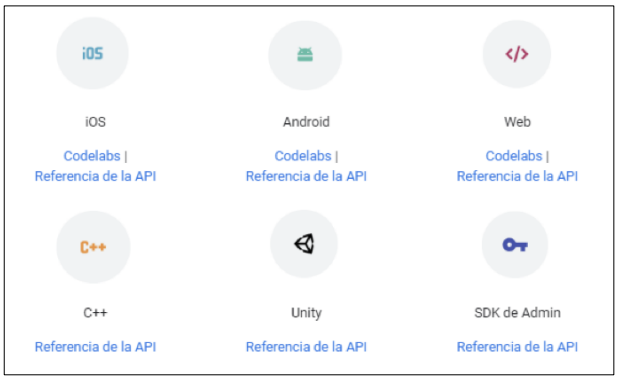


Figura 9: Plataformas y lenguajes soportados por Firebase

Fuente: (Andrade, 2022)

2.10 Protocolo de Tiempo de Red (NTP)

Es una herramienta esencial en sistemas informáticos que requieren sincronización precisa de relojes a través de la red. Este protocolo facilita la coordinación temporal entre dispositivos distribuidos, asegurando que mantengan la hora correcta y estén alineados temporalmente. (Redes Zone, 2023)

Su principal objetivo radica en garantizar que los dispositivos dentro de una red, mantengan un tiempo preciso y estén sincronizados unos con otros. Esta sincronización precisa es crucial para diversas aplicaciones, como la programación de tareas, la autenticación basada en el tiempo y la generación precisa de registros. (Redes Zone, 2023)

2.11 Dispositivos para el desarrollo de prototipos IoT

2.11.1 Kit de desarrollo “ESP32 DevKit V1”

Esp32 es una familia de microcontroladores de la empresa Espressif Systems, es muy superior en capacidades a un Arduino UNO. Por ejemplo, con un dispositivo ESP32 en su versión de dos núcleos se podría realizar un proyecto que intercambie información a la nube

y en simultáneo administre datos de un sensor de forma precisa, situación que se pretende demostrar en el proyecto presentado.
Los ESP32 poseen un alto nivel de integración. En su pequeño encapsulado se incluyen:

- Interruptores de antena.
- Amplificadores de potencia.
- Amplificadores de recepción de bajo ruido.
- Filtros y módulos de administración de energía.

Figura 10: Kit de desarrollo ESP32 DevKit v1

2.11.2 Sensor de humedad relativa y temperatura DHT22

El módulo DHT22 es un sensor digital de temperatura y humedad relativa de bajo costo y excelente rendimiento. El cual es conformado por un sensor capacitivo de humedad y un termistor que sirve para la medición del aire circundante, el cual muestra los datos mediante el envío de una señal digital en el pin de datos del módulo. Es utilizado en aplicaciones que se utilizan para controlar la temperatura, aire acondicionado, monitoreo ambiental en agricultura, estaciones meteorológicas, entre otros.

Figura 11: Sensor de temperatura y humedad DHT22

Fuente: (Unit Electronics, 2018)

2.11.3 Sensor analógico de humedad del suelo

El funcionamiento del sensor se basa en medir la resistencia entre 2 electrodos insertados dentro del suelo, la resistencia entre los electrodos dependerá de la humedad del suelo, por lo que si el suelo tiene un alto nivel de humedad resultar tener resistencia muy baja (corto circuito) y cuando el suelo se encuentre seco la resistencia será muy alta.

Figura 12: Sensor de humedad de dos electrodos resistivos

2.11.4 Sensor de luz (LDR)

Por sus siglas en inglés (Light Depend Resistor) o fotorresistor es una resistencia eléctrica la cual varía su valor en función de la cantidad de luz que incide sobre su superficie. Cuanto mayor sea la intensidad de luz que incide en la superficie del LDR o fotorresistor menor será su resistencia y en cuanto menor sea la luz que incida sobre este mayor será su resistencia.

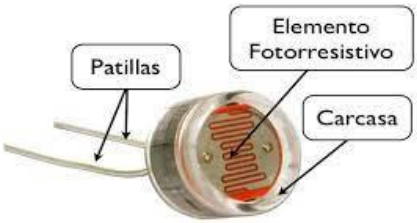


Figura 13: Fotorresistencia

2.11.5 Blender 3D

Es un programa de código abierto que se utiliza comúnmente para la creación, animación, modelado, renderizado, simulación y edición de gráficos tridimensionales, se puede utilizar en una gran variedad de aplicaciones, entre ellas está el diseño de productos para su posterior producción ya sea mediante impresión 3D, corte laser o cualquier otra técnica conocida. (Blender.org, 2024)

2.12 Framework Ionic

Es un SDK de front end de código abierto que sirve para el desarrollo de aplicaciones hibridas, se basa en tecnologías web (HTML, CSS y JS). Es decir, que permite desarrollar Aplicaciones para IOS, Android y web, desde una única base de código. (Agüero, 2021)
Se integra con los principales frameworks de frontend, como Angular, React y Vue, aunque también se puede usar Vanilla JavaScript. (Agüero, 2021)

2.12.1 Características

- Permite desarrollar y desplegar aplicaciones hibridas, que pueden funcionar en varias plataformas y todo ello con una única base de código.
- Ofrece un diseño limpio, sencillo y funcional.
- Emplea Capacitor (o Córdova) para implementar de forma nativa o se ejecuta en el navegador como una aplicación web progresiva.
- Está desarrollado sobre tecnologías web (HTML, CSS y JavaScript)
- Se puede usar con otros frameworks frontend populares, como Angular, React y Vue.

2.12.2 Ventajas y desventajas

En la tabla 4 se indican las ventaja y desventajas de desarrollar un sistema usando Ionic Framework.

Tabla 4: Ventajas y desventajas de usar Ionic framework

Ventajas	Desventajas
Al basarse en tecnologías web, los desarrolladores no tienen que aprender una nueva tecnología para utilizar Ionic.	Peor rendimiento que las aplicaciones nativas.
Ionic se integra con los frameworks comúnmente usados para trabajar, Angular, React y Vue. Además, se integra con numerosas herramientas y plugins.	Cada vez que se requiere acceder a una funcionalidad nativa se debe recurrir a un plugin esto quiere decir que hay una dependencia a los plugins
Favorece a una mayor productividad de los desarrolladores reduciendo costes de desarrollo.	Aplicación más pesada que una nativa, esto implica escribir mucho Código y agregar librerías, complementos y dependencias
Hace más rápido y sencillo el diseño de interfaces de usuario. Se puede ir eligiendo elementos UI predeterminados de su librería de componentes.	No sirve para aplicaciones de carga gráfica y esto quiere decir que no se pueden desarrollar juegos o aplicaciones de gran potencia en gráficos.
Es de código abierto y cuenta con una comunidad activa.	Tiene algunos errores de compilación

Fuente: (Agüero, 2021)

2.13 Framework Angular

Es un framework de ingeniería de software de código abierto construida sobre TypeScript y se utiliza para crear aplicaciones web escalables. Google se encarga del mantenimiento y constantes actualizaciones de mejoras para este framework. (Gonzalves, 2021)
El framework es una creación de los ingenieros de Google, Misko Hevery y Adam Abrons. Google lanzó oficialmente la primera versión, AngularJS, en 2012, y la ha mantenido desde entonces. (Ana, 2023)

2.13.1 Características

Algunas de las principales características de angular que (Coppola, 2022) muestra son:

- Uso de DOM regular.
- Enlace de datos o data binding.
- Compatibilidad movil y de escritorio.
- Velocidad y rendimiento.
- Enlace bidireccional de datos.
- Uso del framework Jasmine para hacer pruebas.

2.14 Metodología Cascada “Waterfall”

Es un enfoque clásico en el desarrollo de software que describe un método de desarrollo lineal y secuencial. Consta de cinco a siete fases, cada fase está definida por diferentes tareas

y objetivos, por lo que la totalidad de las fases describe el ciclo de vida del software o aplicación hasta su entrega. Una vez finalizada una fase, sigue el siguiente paso de desarrollo y los resultados de la fase anterior pasan a la siguiente fase. (Royce, 2019)

2.14.1 Ventajas y desventajas de la metodología "waterfall"

En la tabla 4 se indican las ventajas y algunas desventajas de utilizar la metodología de cascada para desarrollar un sistema.

Tabla 5: Ventajas y desventajas de usar la metodología de cascada (waterfall)

Ventajas	Desventajas
El modelo es simple y fácil de usar.	Si encuentra un error de requisito o necesita cambiar algo, su proyecto debe iniciarse desde el principio con un nuevo código.
Como la metodología es bastante rígida, es fácil de administrar porque cada fase consta de entregables específicos.	Cuando su producto está en la etapa de prueba, no es fácil volver atrás y cambiar algo que no está claro o no se ha formulado bien en la fase inicial.
Las fases no se superponen. Se ejecutan y se completan una a la vez.	No es una buena idea usarlo para proyectos largos con requisitos complejos e imprecisos.
Las metodologías de desarrollo de software en cascada son buenas para proyectos que contienen requisitos claros.	El método no es apropiado para los proyectos en los que se sabe desde inicio que hay muchas probabilidades que los requisitos cambien.
El proceso es predecible.	Los clientes pueden no estar satisfechos con el producto entregado.

Fuente: (Stsepanets, 2023)

2.14.2 Fases de la metodología “waterfall”

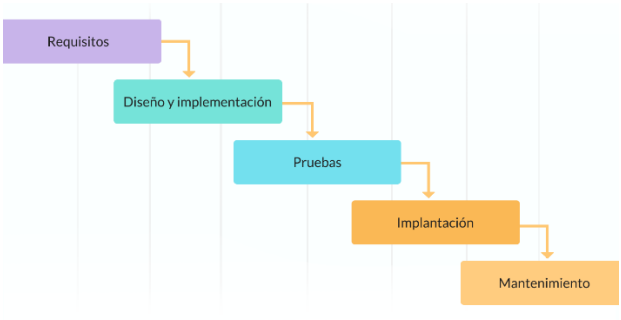


Figura 14: Esquema de la metodología “Waterfall”

Fuente: (Royce, 2019)

CAPÍTULO III METODOLOGÍA

Metodología

Con la presente investigación se creó un prototipo para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil desarrollada con Ionic framework, esto proporcionó una ayuda a los usuarios para que puedan recolectar datos precisos de los parámetros ambientales y el entorno de cultivo.
Este trabajo de investigación se enfocó en un análisis cuantitativo, porque se obtuvieron datos medibles sobre el rendimiento de la aplicación móvil que fue desarrollada a través del modelo de calidad FURPS.

3.1 Tipo de investigación

Aplicada

Porque de acuerdo con su definición, el estudio pretendió solucionar un problema de la vida cotidiana aplicando o utilizando conocimientos desde una o más áreas especializadas con el propósito de implementarlos de forma práctica para satisfacer necesidades o proporcionar soluciones a sectores sociales o productivos como es el caso del monitoreo de cultivos de tomate en huertos urbanos.

3.2 Diseño de investigación

Bibliográfico

Porque la información científica requerida se encuentra en artículos, libros, sitios web relacionados con el tema y revistas de gran utilidad para sustento del proyecto de investigación y fue un punto inicial en el desarrollo.

3.3 Técnicas de recolección de datos

Observación

Con el objetivo de recolectar información sobre los parámetros a evaluar dentro del proyecto de investigación se observaron los datos que los sensores del prototipo muestran, mismos que a través de un dispositivo ESP32 a la base de datos donde la información fue procesada y enviada a la aplicación móvil que se encarga de administrarlos.

3.4 Instrumentos de recolección de datos

Los instrumentos de recolección de datos dentro del tema de investigación planteado son los sensores de Humedad del suelo, temperatura ambiente y luminosidad que tienen la función principal de obtener datos precisos de los fenómenos estudiados.

3.5 Población de estudio y tamaño de muestra

3.5.1 Población de estudio

De acuerdo con el tipo de investigación propuesta, la población es infinita. Se utilizará la herramienta JMeter para obtener datos de acuerdo con las herramientas especificadas por el modelo de calidad FURPS.

3.6 Identificación de variables

- **Variable independiente**
  Aplicación móvil
- **Variable dependiente**
  Fiabilidad de la aplicación móvil
- **Operacionalización de variables**

Tabla 6: Operacionalización de variables

PREGUNTA DE INVESTIGACIÓN	TEMA	OBJETIVOS	VARIABLES	CONCEPTUALIZACIÓN	Dimensión	INDICADORES
En base al modelo de calidad FURPS, ¿Cómo la implementación de un prototipo de IoT con almacenamiento en la nube y soporte en una aplicación móvil usando Ionic es fiable en la agricultura inteligente?	Prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic.	**a. Objetivo general** Desarrollar un prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic. **b. Objetivos específicos** -Investigar las aplicaciones de IoT para el monitoreo de cultivos. -Implementar el prototipo IoT usando una aplicación móvil con Ionic para el monitoreo del cultivo de tomate. -Evaluar la fiabilidad de la aplicación móvil para el monitoreo del cultivo de tomate usando el modelo de calidad FURPS.	**Independiente:** Aplicación móvil.	Aplicación de software que se instala en dispositivos móviles o tabletas para ayudar al usuario en una labor concreta.	-Módulos -Requisitos funcionales .	**Independiente:** -El número de módulos del aplicativo. -Número de requisitos realizados.
			Dependiente: Fiabilidad de la aplicación móvil.	La fiabilidad agrupa los requerimientos que tienen relación con la solidez y robustez de un sistema mediante su ejecución	•Recuperabilidad •Precisión •Predicción	**Dependiente:** % de recuperabilidad. % de precisión. % de predicción.

3.7 Métodos de análisis, y procesamiento de los datos

Una vez obtenida la información de las observaciones, se hizo un análisis para definir una orientación sobre el desarrollo de la aplicación móvil, considerando ciertos criterios para el correcto funcionamiento de este, en el que se consideraron recomendaciones sobre la presentación de contenidos y el diseño.

Después se evaluó la fiabilidad del aplicativo según los criterios del modelo de calidad FURPS.

- Recuperabilidad
- Predicción
- Precisión

3.8 Desarrollo de la aplicación móvil usando la metodología waterfall

3.8.1 Análisis

El análisis en la metodología de cascada implica la recopilación y documentación de los requisitos, necesidades del usuario, y la identificación de las funciones y características que el software debe tener para satisfacer esas necesidades. Este proceso se realiza mediante entrevistas con los usuarios y otras partes interesadas, la revisión de la documentación existente, análisis y pruebas de sistemas existentes similares.

Definición de los requisitos del software

Requerimientos funcionales

Se determinan en base a las funcionalidades que la aplicación en desarrollo requiere (ver Tabla 7.)

Tabla 7: Requerimientos funcionales

Identificador	RF01
Nombre	Registro de usuario
Características	Registro a través del correo electrónico
Descripción	El usuario debe ingresar un correo electrónico valido para el registro.
Entradas	
Correo electrónico Nombre Contraseña	
Salida	
Mensaje de registro exitoso	
Identificador	RF02
Nombre	Ingreso al aplicativo
Requerimiento que lo Utiliza o Especializa	RF01
Características	Inicio de sesión proporcionando sus datos
Descripción	El usuario debe iniciar sesión para poder acceder al contenido y herramientas de la aplicación.
Entradas	
Correo electrónico Contraseña	
Salida	
página principal de la aplicación	
Identificador	RF03
Nombre	Vista en tiempo real (cada 20 segundos) de los datos del sistema
Requerimiento que lo Utiliza o Especializa	RF03
Características	Se observará información detalla relacionada al servicio
Descripción	Al ingresar en la página de inicio se mostrarán los datos de la planta (nombre, fecha, humedad del suelo, temperatura ambiente, humedad relativa y luminosidad.
Entradas	
ninguna	

Salida	
Datos en tiempo real de los sensores	
Identificador	RF04
Nombre	Menú
Requerimiento que lo Utiliza o Especializa	RF03
Características	Sección menú tiene información del usuario con opciones de: cerrar sesión, acerca de
Descripción	El usuario podrá ingresar a la sección del menú en donde podrá seleccionar las opciones mostradas.
Entradas	
Solicitud táctil	
Salida	
Ingreso al menú de opciones	
Identificador	RF05
Nombre	Cerrar sesión
Requerimiento que lo Utiliza o Especializa	RF02
Características	Al iniciar sesión, dentro de la interfaz se podrá cerrar sesión de manera segura
Descripción	Una vez dentro de la aplicación el usuario tendrá la opción cerrar sesión de manera segura, dirigiéndolo a la vista principal de donde puede iniciar sesión nuevamente si lo desea.
Entradas	
Ninguna	
Salida	
Ninguna	
Identificador	RF07
Nombre	Verificación para recuperar o cambiar contraseña
Requerimiento que lo Utiliza o Especializa	RF02
Características	Verificación para recuperar contraseña a través del correo electrónico.
Descripción	El usuario debe ingresar el correo electrónico que utilizó al momento de registrarse para obtener un código de acceso que le permitirá recuperar o cambiar la contraseña.
Entradas	
Correo electrónico Código de verificación	
Salida	
Correo del sistema con código de verificación	

Requerimientos no funcionales

Describen criterios del funcionamiento general del sistema, estos requisitos comprenden características de seguridad, disponibilidad, compatibilidad, escalabilidad, etc. (ver Tabla 8)

Tabla 8: Requerimientos no funcionales

Identificador	RNF01
Nombre	Portabilidad
Características	Sera desarrollado con Ionic Framework

Descripción	El aplicativo será desarrollado para móviles con sistema operativo Android.
Prioridad	**Alta**
Identificador	RNF02
Nombre	Instalación del aplicativo.
Características	Fácil instalación en el dispositivo móvil.
Descripción	El aplicativo podrá ser instalado mediante su respectivo APK (Android Package).
Prioridad	**Alta**
Identificador	RNF03
Nombre	Ingreso a la Aplicación Móvil.
Características	Facilidad de Uso.
Descripción	El sistema debe permitir al usuario ingresar su usuario y contraseña.
Prioridad	**Alta**
Identificador	RNF04
Nombre	Tiempo de respuesta
Características	Tiempo de respuesta a la hora de usar el aplicativo.
Descripción	El tiempo de carga de las interfaces no debe superar los 5 segundos.
Prioridad	**Alta**
Identificador	RNF05
Nombre	Interfaz del sistema
Características	Interfaz amigable para el usuario.
Descripción	El sistema presentara una interfaz de usuario sencilla e interactiva para que sea de fácil manejo dentro de la aplicación móvil.
Prioridad	**Alta**
Identificador	RNF06
Nombre	Desempeño
Características	Buena respuesta del aplicativo hacia la acción que realice el usuario.
Descripción	El sistema deberá alojarse en un servidor eficiente que pueda manejar gran concurrencia de datos en ciertos periodos.
Prioridad	**Alta**
Identificador	RNF07
Nombre	Disponibilidad
Características	Conexión estable al servidor
Descripción	La disponibilidad del sistema debe ser continua con un nivel de servicio eficiente para los usuarios y así evitar problemas de conexión.
Prioridad	**Alta**
Identificador	RNF08
Nombre	Arquitectura
Características	Aplicación escalable para futuras mejoras.
Descripción	El aplicativo debe permitir futuras mejoras sí es el caso.
Prioridad	**Media**

Identificador	RNF09
Nombre	Diseño adaptable
Características	Adaptación de la interfaz a diferentes resoluciones.
Descripción	La aplicación debe adaptarse a diferentes resoluciones de pantalla en los distintos dispositivos son sistema operativo Android.
Prioridad	**Alta**

Cronograma de desarrollo

Posterior a analizar los requerimientos de la aplicación móvil se diseñó la planificación con los plazos para desarrollar el proyecto indicando el tiempo en meses de las actividades, así como se muestra en la tabla 9.

Tabla 9: Cronograma de desarrollo

No.	ACTIVIDAD	MES					
		Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6
1	**Análisis**						
1.1	Análisis de los requisitos de Software	X					
1.1.1	Requerimientos funcionales	X					
1.1.2	Requerimientos no funcionales	X					
2	**Diseño**						
2.1	Diseño de procesos	X					
2.2	Diseño de casos de uso		X				
2.3	Diseño de la interfaz		X				
2.4	Diseño de la base de datos MongoDB		X				
2.5	Diseño del Hardware			X			
3	**Codificación**						
3.1	Codificación del microcontrolador				X		
3.2	Desarrollo del prototipo electrónico				X		
3.3	Desarrollo frontend (Ionic, Angular)					X	
3.4	Desarrollo backend (Express.js)					X	
4	**Prueba**						
4.1	Pruebas del sistema y de integración						X
4.2	Preparación del entorno de pruebas en JMeter						X
4.3	Evaluar la fiabilidad de la aplicación móvil						X

3.8.2 Diseño

Durante la fase de diseño, se llevó a cabo una meticulosa concepción de la arquitectura del software, la cual se fundamentó en un análisis exhaustivo del tipo de proyecto en cuestión. Como resultado, se procedió a desarrollar una arquitectura IoT de tres capas. En primer lugar, la capa de percepción, que constituye la capa física del sistema, integra una variedad de sensores diseñados para detectar y recopilar datos sobre el entorno circundante, abarcando parámetros físicos relevantes. En segundo lugar, la capa de red asume la responsabilidad de establecer conexiones con dispositivos inteligentes, así como con dispositivos de red y servidores. Esta capa no solo facilita la transmisión de datos desde los sensores, sino que también se encarga del procesamiento inicial de los mismos. Por último, la capa de aplicación se dedica a ofrecer servicios específicos de la aplicación al usuario final. Esta capa se configura de manera que pueda adaptarse eficazmente a diversas aplicaciones de IoT, tales como sistemas para hogares inteligentes, gestión de cultivos en huertos inteligentes, soluciones urbanas inteligentes y sistemas de salud inteligente.

DISEÑO DE PROCESOS

El diseño de procesos se refiere al proceso de planificación y creación de sistemas y procedimientos para realizar actividades eficientes y efectivas. Estos diseños se centran en definir como se realizarán las actividades dentro de una organización para lograr objetivos específicos. Proporcionan una estructura clara y coherente.

A continuación, en la figura 15 se describe el proceso de inicio (Login) que permite ingresar a la interfaz principal de la aplicación luego de colocar las credenciales.

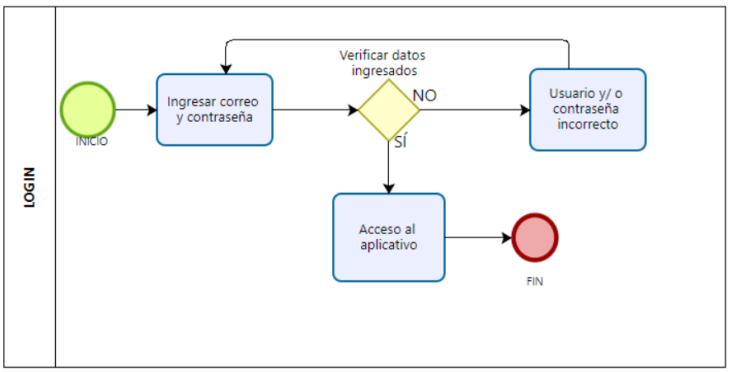


Figura 15: Diagrama de proceso Login

A continuación, en la figura 16 se muestra el proceso de “registro de usuario” en el cual sí el usuario aún no se encuentra registrado deberá llenar los campos necesarios para su posterior registro en la app móvil.

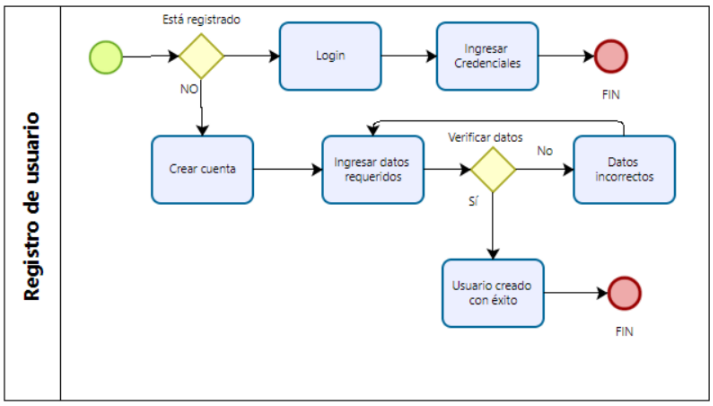


Figura 16: Diagrama de registro de nuevo usuario

En la figura 17, el proceso para "recuperar contraseña". Se deberá colocar un correo electrónico con el que se ha registrado para a continuación recibir un link para realizar el cambio de una contraseña nueva.

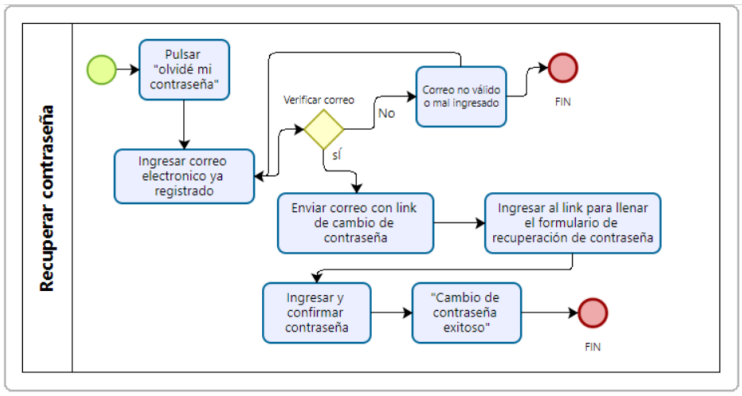


Figura 17: Diagrama de proceso para recuperar contraseña

La figura 18 muestra el diagrama del proceso de conexión con el dispositivo del sensor que recogerá los parámetros ambientales.

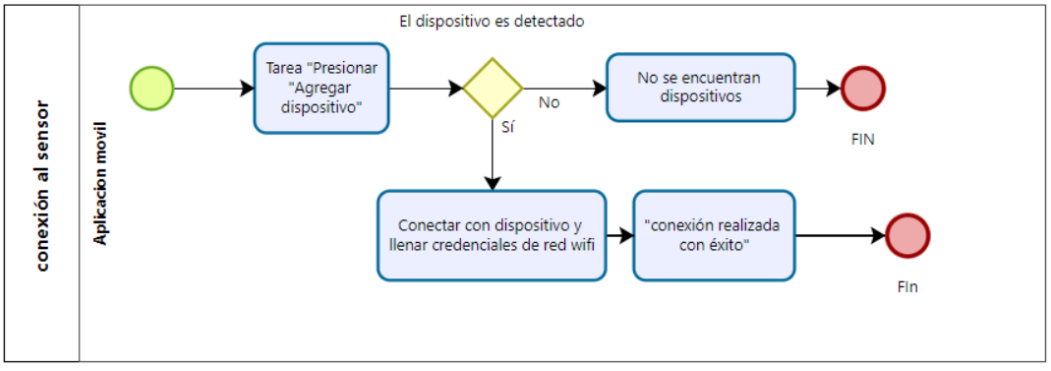


Figura 18: Diagrama de proceso para conexión con sensor IoT

DISEÑO DE CASOS DE USO

Los casos de uso representan una técnica primordial en la ingeniería de software, orientada a la captura y descripción de las interacciones entre los actores, que en este contexto son los usuarios, y el sistema de software en cuestión. Estos casos se erigen como una modalidad de documentación que detalla los requisitos funcionales del sistema desde la óptica del usuario. Su utilidad radica en su capacidad para cristalizar y formalizar los requisitos del sistema, asegurando así su comprensión precisa por parte de las partes interesadas, lo que a su vez contribuye al desarrollo coherente y exitoso del proyecto.

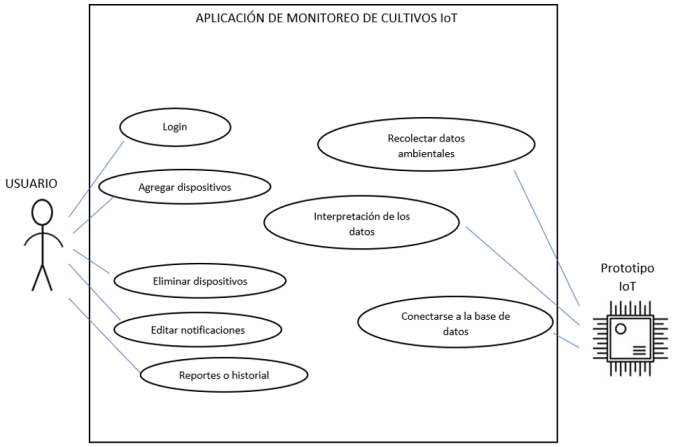


Figura 19: Casos de Uso

Diagrama de la arquitectura

La aplicación móvil desarrollada es arquitectura Cliente/Servidor, implica una comunicación bidireccional entre el dispositivo IoT que va a recoger y enviar los datos de los sensores y el servidor backend para solicitar datos, y una comunicación entre la aplicación móvil con el servidor backend para solicitar y mostrar los datos de los parámetros recogidos.
A continuación, se muestra en la figura 20 la arquitectura cliente-servidor del proyecto.

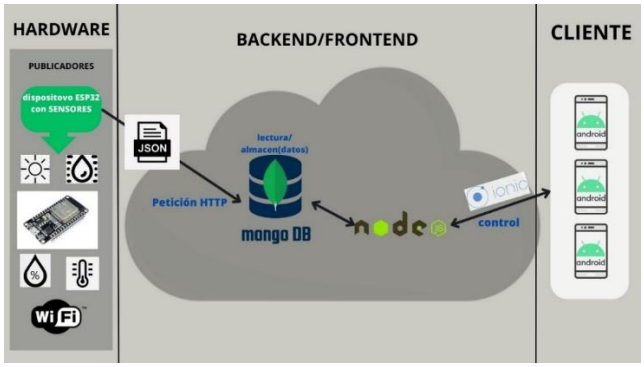


Figura 20: Arquitectura Cliente-Servidor

Implementación del diseño de la interfaz

El diseño de la interfaz se refiere al proceso de crear una experiencia visual y funcional que permita a los usuarios interactuar de manera efectiva y eficiente con un producto o

sistema. Esto se puede aplicar en un amplio espectro de productos que incluyen aplicaciones móviles, sitios web, software de escritorio, dispositivos electrónicos y más.

Se ha desarrollado un diseño de inicio de sesión básico, donde el usuario debe ingresar su identificador de usuario, que en este caso corresponde a su dirección de correo electrónico, junto con la contraseña asociada a su cuenta. Una vez ingresados estos datos, el usuario puede activar la función de inicio de sesión al presionar el botón correspondiente. Además, en la misma interfaz se incluyen opciones adicionales, como la posibilidad de recuperar la contraseña y de crear una nueva cuenta.

Figura 21: Interfaz de Login

Elaborado por: Lennin Manrique

En la interfaz de registro de la figura 22 se escribe un nombre de usuario, colocar un correo electrónico valido se escribe una contraseña, se vuelve a escribir la contraseña para confirmar y pulsar el botón “registrarse” tras llenar el formulario de registro.

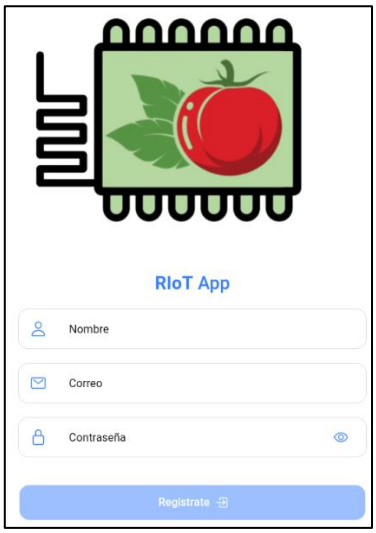


Figura 22: Interfaz de registro

La figura 23 indica la interfaz para recuperar la contraseña. Se requiere ingresar el correo que está asociado a la cuenta a la cual se ha olvidado la contraseña, un correo electrónico con un enlace que dirige a una ventana de cambio de contraseña se envía a la bandeja principal del correo asociado.

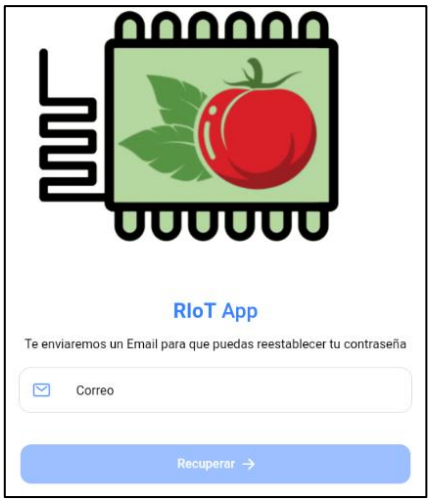


Figura 23: Interfaz de reestablecer contraseña

Se diseño la interfaz principal del usuario luego de ingresar, se mostrarán cuatro cards en los cuales cada uno indica su respectivo parámetro de lectura por ejemplo la temperatura y humedad del suelo, además de la sección de información de la planta. La parte inferior derecha despliega el menú donde se encuentra la opción de reportes diarios.

Figura 24: Interfaz de inicio de la aplicación

La aplicación cuenta con una página de reportes diarios de las lecturas mostradas en graficas de barras que muestran los promedios por hora de los parámetros ambientales como se muestra en la figura siguiente (ver figura 25).

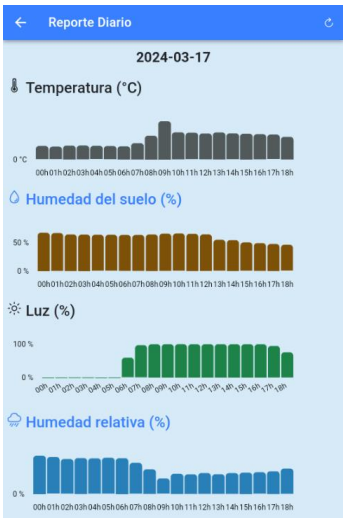


Figura 25: Interfaz de reportes diarios

DISEÑO DE BASE DE DATOS

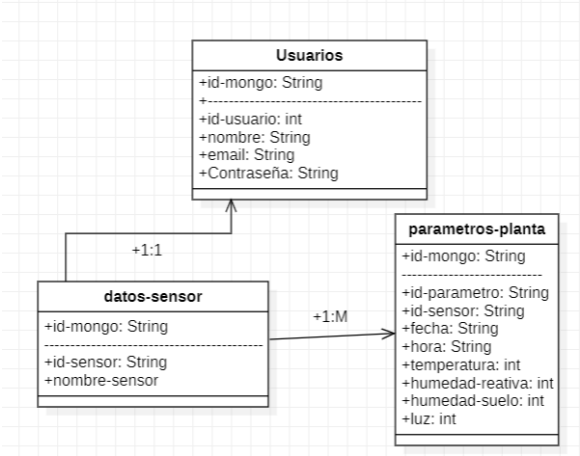


Figura 26: Diseño de la base de datos

Diseño del hardware

La figura 27 muestra el diseño del diagrama de conexiones del circuito que conforma el prototipo físico de monitoreo el cual consta de un microcontrolador ESP32 en su versión uno de la marca Espressif inc, resistencias, fotorresistencias, un módulo comercial DHT22 y un sensor de humedad basado en la técnica de divisores de voltaje.

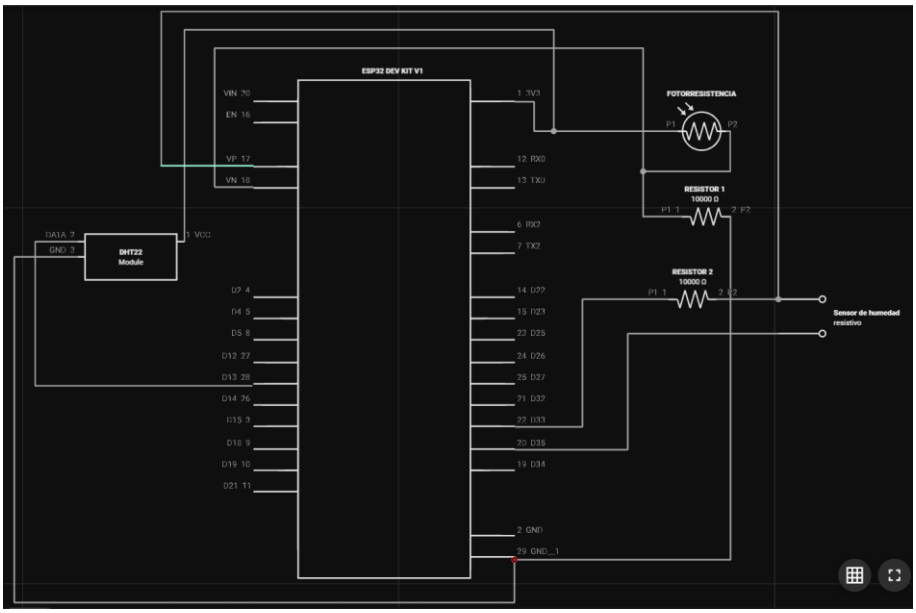


Figura 27: Diagrama eléctrico del prototipo

En la tabla 10 se muestran los parámetros de medición del módulo de humedad relativa y temperatura ambiente DHT22 y sus especificaciones técnicas.

Tabla 10: Especificaciones técnicas DHT22

Modelo	**DHT22**
Voltaje de trabajo	3.3-6V DC
Señal de salida	Señal digital vía single-bus
Elemento sensor	Capacitor de polímero
Rango de operación	Humedad 0-100% RH (humedad relativa); Temperatura -40~80 Celsius
Exactitud	Humedad +-2% RH (máximo +-5% RH); Temperatura <+-0.5 Celsius
Resolución o sensibilidad	Humedad 0.1% Temperatura 0.1 Celsius
Repetibilidad	Humedad +-1% RH Temperatura +-0.2 Celsius
Histéresis de humedad	+- 0.3% RH
Estabilidad a largo plazo	+- 0.5% RH/año
Periodo de lectura	Promedio: 2 segundos
Intercambiabilidad	Totalmente intercambiable
Dimensiones	Tamaño pequeño: 14*18*5.5 mm; Tamaño grande: 22*28*5mm

A continuación, se muestran las especificaciones técnicas de la fotorresistencia o sensor de luz LDR. Una fotorresistencia es un componente físico que es capaz de cambiar su conductividad eléctrica según la cantidad de luz que llegue a su superficie.

Tabla 11: Especificaciones técnicas LDR 12mm

Modelo	LDR
Rango de trabajo en temperatura	-30 a +70 Celsius
Voltaje máximo	250 voltios
Disipación de potencia máxima	200mW
Resistencia a oscuras	>1 millón de ohms
Longitud de onda de detección máxima	540nm (similar al ojo humano)
Dimensiones	12 milímetros

Para el sensor de humedad del suelo se utilizó un divisor de voltaje o divisor de tensión que consiste en dos resistencias conectadas en serie entre la fuente de alimentación (3.3 voltios) y la tierra (gnd). Este sensor de humedad del suelo suele consistir en dos electrodos metálicos que se insertan en el suelo. Cuando el suelo está húmedo, actúa como un conductor y reduce la resistencia eléctrica entre los electrodos y por otro lado cuando el suelo está seco, su resistencia eléctrica aumenta y así se puede determinar el nivel de humedad del suelo.
El divisor de tensión se usa junto con el sensor de humedad del suelo para convertir el cambio de resistencia del suelo en una señal eléctrica que pueda medir y usar el microcontrolador.
El microcontrolador mide la tensión de salida del divisor de tensión y la convierte en un valor de humedad del suelo mediante un proceso de calibración y mapeo de datos.
La figura 28 muestra una representación gráfica de lo antes descrito.

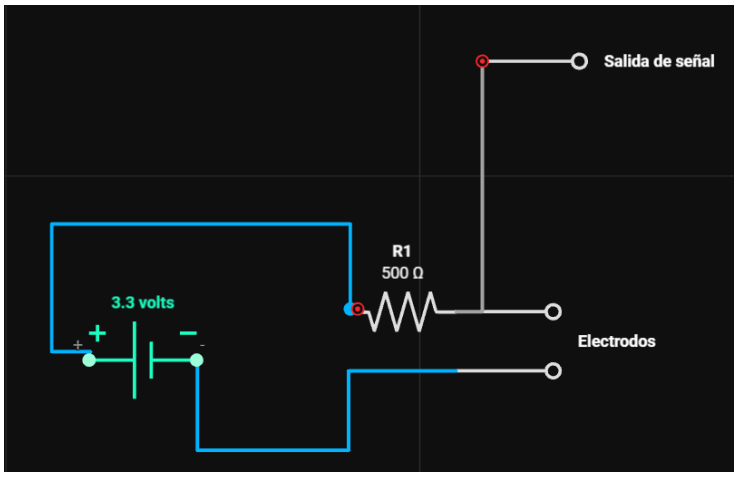


Figura 28: Sensor de humedad resistiva (divisor de voltaje)

Para diseñar la parte física del prototipo se utilizó el modelador 3D gratuito Blender. El diseño está basado en la rama de una planta, tiene el espacio suficiente en su interior para que los componentes encajen sin problemas.
En la figura 29 se muestran las partes del diseño del prototipo, consta de dos secciones separadas que se pueden unir luego de haber colocado los componentes en su interior.

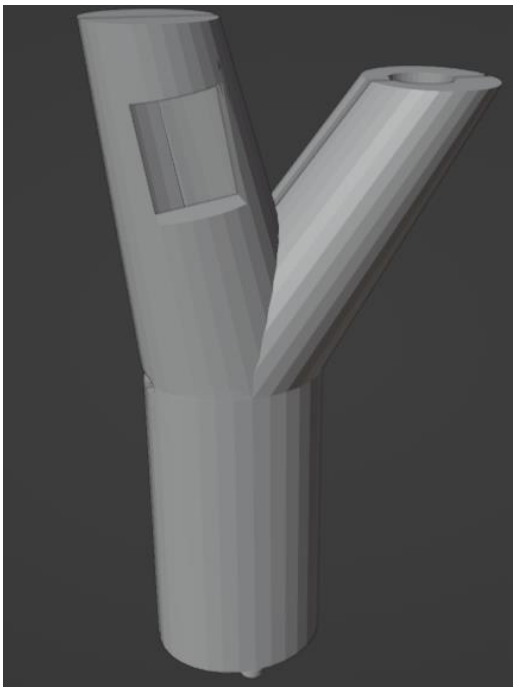


Figura 29: Diseño 3D prototipo de monitoreo

3.8.3 Codificación

Codificación del microcontrolador

Para desarrollar el sistema se ha utilizado el lenguaje de Typescript para el frontend y backend utilizando el editor de código fuente Visual Studio Code, para la codificación del hardware ESP32 se utiliza la plataforma de Arduino que se programa con un lenguaje propio basado en el lenguaje de programación de alto nivel Processing, lo que significa que es similar a C++ y además posee una arquitectura IoT de 4 capas: capa sensorial, capa de transporte, capa de procesamiento de datos y la capa de aplicación GUI.
Para la codificación del microcontrolador se requiere de la instalación de las siguientes herramientas:

- Arduino IDE 2.2.1 de escritorio
- Gestor de placas para poder programar el microcontrolador ESP32 (es necesaria la URL oficial). La URL es la siguiente: https://dl.espressif.com/dl/package_esp32_index.json
- Librería actualizada <HTTPClient.h>.
- Librería <ArduinoJson.H> en su última versión.
- Librería <WiFi.h> (viene incluida en el paquete de la placa del ESP32).
- Liberia predefinida que sirve para usar el módulo sensor de temperatura y humedad relativa <DHT.H>.
- Librería <time.h> para la gestión de la hora y la fecha.

a) Función de hora y fecha

El siguiente código muestra la función que se utilizó dentro de microcontrolador para poder obtener le fecha y hora de un servidor NTP mediante su API. Recoge la fecha y hora actual del sistema para posteriormente convertir esos dos datos en formato de documento tipo JSON.

```
void getTime(){

  WiFiClient client;
  HTTPClient http;
  String timeS = "";

  http.begin(client,"http://worldtimeapi.org/api/timezone/America/Guayaquil");
  int httpCode = http.GET();
  if(httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY)
  {
    String payload = http.getString();
    int beginS = payload.indexOf("datetime");
    int endS = payload.indexOf("day_of_week");
     //Serial.println(payload);
    String timeS = payload.substring(beginS + 11, endS -25 ); //3
    Serial.print("fecha: ");
    Serial.println(timeS);
    jsonDoc["fecha"] = timeS;
    //hora...
    String timeShora = payload.substring(beginS + 22, endS -9);
    Serial.print("hora: ");
    Serial.println(timeShora);
    jsonDoc["hora"] = timeShora;
    //jsonDoc["hora"] = ;
  }
  Serial.println("-----------------------------------------------");
```

Figura 30: función fecha y hora

b) Función para leer datos del módulo DHT22

Para poder usar el módulo de temperatura y humedad relativa va a ser necesario el uso de una librería diseñada por el fabricante. El siguiente código muestra cómo se obtienen los datos del sensor de temperatura y humedad relativa, además, se usa un condicionante "while" en caso de que el dispositivo no estés funcionando correctamente.

```
//funcion para lectura de modulo DHT22
void leerdht1 (){
  float t1 = dht1.readTemperature();
  int h1 = dht1.readHumidity();
  while (isnan(t1) || isnan(h1)){
   Serial.println("lectura fallida en el sensor dht22, repitiendo lectura.....");
   delay(2000);···
  }
  Serial.print("Temperatura DHT22: ");
  Serial.print(t1);
  Serial.println(" °C.");
  Serial.print("Humedad DHT22: ");
  Serial.print(h1);
  Serial.println(" %.");
  Serial.println("");
  //json documento
  jsonDoc["temperatura"] = t1;
  jsonDoc["humedad-relativa"] = h1;

}
```

Figura 31: función DHT22

c) Función de lectura para sensor de luz LDR

Para la lectura del sensor de luz se usa una célula fotoconductora o más comúnmente conocida como LDR. EL siguiente código se encarga de leer los datos recogidos en el divisor de voltaje formado por una resistencia de 10k ohm y el fotorresistor, esos datos son mapeados para poder obtener un porcentaje de luz recogida.

```
// funcion para leer el LDR(sensor de luz)
void readLdr (){
  int ldr_val = analogRead(pin2);
  ldr_val = map(ldr_val,0,4095,0,100);
  // se imprime en el monitor serie para pruebas
  Serial.print("luminosidad: ");
  Serial.print(ldr_val);
  Serial.println(" %LUX");
  Serial.println("");
  //JSON DOC
  jsonDoc["luz"] = ldr_val;
}
```

Figura 32: código de sensor de luz LDR

d) Función de lectura para humedad del suelo

En el código para la lectura del sensor de humedad del suelo se requiere una configuración que aparte de recoger los datos que envía el divisor de voltaje, también, físicamente controle la corrosión rápida de los electrodos que irán insertados en el suelo. Para obtener el porcentaje de humedad del suelo simplemente se realiza un mapeo.

```
// funcion para leer la humedad del suelo
void rdHumSuelo (){

  pinMode(0,OUTPUT);
  pinMode(2,OUTPUT);
  digitalWrite(0,HIGH);
  digitalWrite(2,LOW);
  delayMicroseconds(pausa);
  int humValue = analogRead(pin3);
  digitalWrite(0,LOW);
  digitalWrite(2,HIGH);
  delayMicroseconds(pausa + 108);
  pinMode(0,INPUT);
  pinMode(2,INPUT);
  //mapeo para tipos de suelos y porcentajes
  humValue = map(humValue,0,4095,0,100);
  // se imprime en el monitor serie para pruebas
  Serial.print("humedad suelo: ");
  Serial.print(humValue);
  Serial.println(" %");
  Serial.println("--------------------------");
  //json doc
  jsonDoc["humedad-suelo"] = humValue;
}
```

Figura 33: código del sensor de humedad-suelo

e) Envío de datos a Mongodb Atlas mediante un documento JSON

Luego de recoger los datos de cada sensor, convierten a un formato que sea reconocible por la base de datos de mongoDb que en este caso es JSON y mediante peticiones HTTP POST se envía el documento a la base de datos correspondiente.

```
Serial.println("datos enviados a mongoDB Atlas.......");
// Convertir el objeto JSON a una cadena
String json;

serializeJson(jsonDoc, json);
Serial.println(json);

// Enviar los datos a MongoDB Atlas
HTTPClient http;
http.begin(serverName);
http.addHeader("Content-Type", "application/json");

 int httpResponseCode = http.POST(json);

if (httpResponseCode > 0) {
  Serial.print("Respuesta del servidor: ");
  Serial.println(httpResponseCode);
} else {
  Serial.print("Error en la solicitud HTTP. Código de error: ");
  Serial.println(httpResponseCode);
}
Serial.println("-----------------------------------------------");

delay(20000);
http.end();
}
```

Figura 34: Envío de parámetros ambientales a mongodb Atlas

Posterior a terminar el prototipo de monitoreo y su codificación respectiva se procede al desarrollo de la aplicación móvil que se encarga de gestionar los parámetros que se han enviado al servidor de base de datos.

Codificación de la aplicación móvil

Para poder iniciar con la codificación de la aplicación móvil es necesario instalar las siguientes herramientas en el ordenador.

- Visual Studio Code (Editor más usado para desarrollo de apps web y móviles entre otras)
- Mongodb Compass (gestor gráfico de base de datos NoSQL de mongodb inc)
- Node.js
- Ionic Framework
- Angular Framework

a) Creación de clúster en Mongodb Atlas para gestionar bases de datos NoSQL desde la nube.

MongoDB ofrece un servicio en la nube llamado MongoDB Atlas que permite a los desarrolladores implementar y administrar de forma fácil clústeres de base de datos de MongoDB en la nube. La siguiente figura muestra el clúster que fue creado para el almacenamiento de los datos que son usados por la aplicación móvil se lo llamó cluster0.

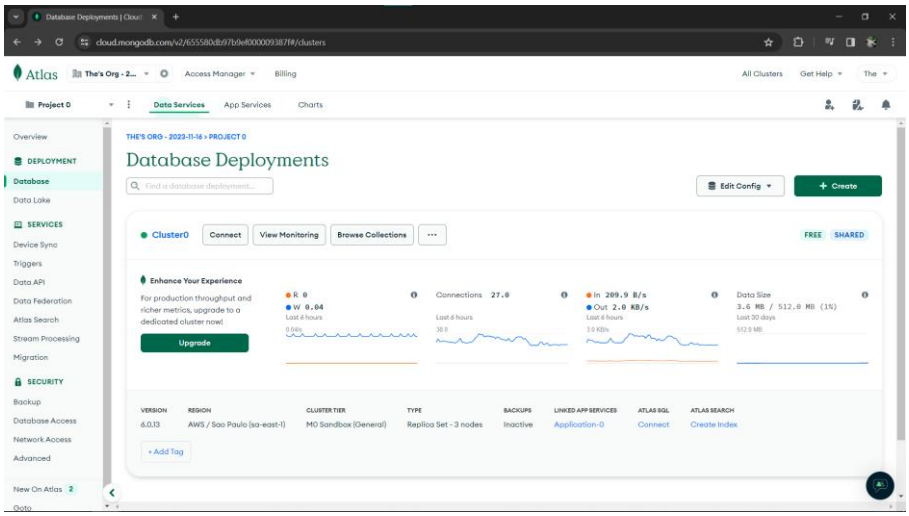


Figura 35: clúster MongoDB Atlas

b) Codificación del servidor

En el archivo index.js del servidor backend, se realizan las configuraciones iniciales para establecer una conexión con la base de datos MongoDB y definir las rutas de la aplicación Express. Una descripción más concreta de lo que se muestra en la figura 36:

- **Importación de módulos:** Se importan los módulos necesarios para el funcionamiento del servidor, incluyendo Express, MongoDB, CORS y Multer (utilizado para el manejo de archivos).
- **Inicialización de Express:** Se crea una instancia de Express para la aplicación.
- **Conexión con MongoDB:** Se utiliza el método MongoClient.connect() para establecer una conexión con la base de datos MongoDB utilizando la cadena de conexión proporcionada. Una vez establecida la conexión, se asigna la instancia de la base de datos a la variable "database".
- **Inicio del servidor:** Se inicia el servidor Express en el puerto especificado.
- **Mensaje de conexión exitosa:** Se muestra un mensaje indicando que la conexión con la base de datos fue exitosa.

```
Ionic2 > RioT-Agro-3 > api > index.js > ...
1  var Express = require("express");
2  var MongoClient= require("mongodb").MongoClient;
3  var cors=require("cors");
4  const multer=require("multer");
5  const { error } = require("console");
6
7  var app=Express();
8  app.use(cors());
9
10 var CONNECTION_STRING="mongodb+srv://lenmanri29:JKnDqGdq8FYR7Zhg@cluster0.snrto53.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0";
11 const PORT = process.env.PORT || 5038;
12 var DATABASENAME="prototipo";
13 var database;
14 app.listen(5038,()=>{
15     MongoClient.connect(CONNECTION_STRING,(error,client)=>{
16         database=client.db(DATABASENAME);
17         console.log("Mongo DB conexion exitosa...");
18     });
19
```

Figura 36: Servidor backend

Posterior a realizar la conexión a la base de datos se escriben los endpoints necesarios que va a requerir la aplicación móvil. EL código completo del servidor se muestra en la sección de anexos como "Código del servidor index.js"

```
app.get('/api/proto/parametros',(request,response)=>{
        database.collection("parametros-planta").find({}).toArray((error,result)=>{
        response.send(result);
    });
})
//obetner el ultimo documento
app.get('/api/proto/ultimaLectura', (request, response) => {
    database.collection("parametros-planta")
    .find({})
    .sort({_id: -1})
    .limit(1)
    .toArray((error,result)=>{
        if (error) throw error;
        response.send(result);
    });
})
//lecturas por fecha//
app.get('/api/proto/lecturaDiaria', (request, response) => {
    // Obtener la fecha del último documento devuelto por el primer endpoint
    database.collection("parametros-planta")
    .find({})
    .sort({_id: -1})
    .limit(1)
    .toArray((error,result)=>{
        if (error) throw error;
        // Extraer los primeros 10 caracteres de la fecha del primer documento devuelto
        const fechaDocumento = result[0].date.substring(0, 10);
        // Consultar la colección para obtener todos los documentos que coincidan con la fecha
        database.collection("parametros-planta")
        .find({
            "date": { $regex: new RegExp("^" + fechaDocumento) } // Utilizar expresión regular para comparar los primeros 10 caracteres
        })
        .toArray((error, resultados) => {
            if (error) throw error;
            response.send(resultados);
        });
    });
});
```

Figura 37: Endpoints del Backend

c) Codificación de la interfaz

La página Home de nuestra aplicación móvil utiliza HTML y el framework Ionic, diseñado para aplicaciones híbridas. Ionic ofrece componentes optimizados para dispositivos móviles, como <ion-header>, <ion-toolbar>, <ion-title>, <ion-buttons>, <ion-content>, <ion-label>, etc.

Se usaron directivas *ngFor de Angular para iterar sobre una matriz de datos llamada parámetro y mostrar información dinámica en la interfaz, incluyendo ID del sensor, fecha, temperatura, humedad del suelo, humedad relativa y luz.

Se implementaron funcionalidades interactivas para dispositivos móviles, como abrir un menú al hacer clic en un botón táctil. Además, utilizamos botones flotantes <ion-fab> para acciones rápidas y accesibles en pantallas táctiles. Se puede encontrar la codificación completa en Anexo 2.

```
<ion-header>
  <ion-toolbar *ngFor="let param of parametro" color="primary">
      <ion-title slot="start">
        Home RIoT
                
        {{param['id-sensor']}}
      </ion-title>
      <ion-buttons slot="end" >
        <ion-button (click)="openMenu()">
          <ion-icon slot="icon-only" name="ellipsis-vertical"></ion-icon>
        </ion-button>
      </ion-buttons>
  </ion-toolbar>
</ion-header>
<!-- ============== contenido ================ -->
<ion-content >

<div class="video-container ">
  <video autoplay loop muted class="video">
    <source src="assets/videoHome.mp4" type="video/mp4" >
  </video>
</div>
<!-- ================== Info Tomate============== -->
<div>
  <ion-item lines="full" class="larger-item"  routerLink="../info-tomate" >

    <ion-avatar slot="start" class="avatar-large">
      <img alt="Solanum lycopersicum" src="assets/pictures/tomatePerfil1.jpg" />
    </ion-avatar>
    <ion-label><b>Solanum lycopersicum </b> <br> Tomate</ion-label>
    <ion-icon slot="end" name="chevron-forward"></ion-icon>
  </ion-item>
</div>
<!-- <p *ngFor="let param of parametro">{{ formatDate(param.date) }}</p> -->
<h3 *ngFor="let param of parametro" class="ion-text-center ion-no-margin">
  <b>{{param['date'] |slice:0:10 }}</b>
</h3>
<ion-grid fixed >
  <ion-row>
    <!-- ==========TEMPERATURA======== -->
    <ion-col size="6">
      <ion-card color="medium"  class="card-temperatura">
        <ion-icon class="icono-grande" name="thermometer-outline"></ion-icon>
        <ion-text  >
          <h1><b>Temperatura</b></h1>
          <h2 *ngFor="let param of parametro" >
            {{param.temperatura | number:'1.2-2'}}°C
          </h2>
        </ion-text>
```

Figura 38: Código página Home.html

El diseño de la página de reporte diario se estructura en dos partes principales: el encabezado y el contenido.

El encabezado, definido dentro del componente <ion-header>, contiene un título descriptivo y dos botones. El primer botón, con la función de retroceder, redirige al usuario a la sección principal de la aplicación (home). El segundo botón, de recarga, permite al usuario actualizar manualmente los datos de las tablas, aunque la lógica ya está configurada para actualizar automáticamente cada hora.

El contenido, encapsulado en <ion-content>, muestra los reportes diarios de los diferentes parámetros (temperatura, humedad del suelo, luz y humedad relativa). Cada tabla incluye un título identificativo y un gráfico de barras generado con la librería “ngx-charts-bar-vertical”. Para garantizar una experiencia de usuario óptima, se emplean directivas estructurales de Angular (*ngIf) para mostrar los gráficos únicamente cuando los datos están disponibles.

La información se presenta de forma visualmente atractiva y comprensible mediante el uso de iconos junto a los nombres de los parámetros, lo que facilita su interpretación por parte de los usuarios. Además, cada tabla cuenta con etiquetas claras para mejorar la comprensión de la información presentada.

```
<ion-header>
  <ion-toolbar color="primary">
    <ion-buttons slot="start">
      <ion-back-button defaultHref="/"></ion-back-button>
    </ion-buttons>
    <ion-title>Reporte Diario</ion-title>
    <ion-buttons slot="end">
      <ion-button>
        <ion-icon slot="" name="refresh"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<!-- ====== reportes ========== -->
<ion-content>
  <ion-text color="dark" class="ion-text-center">
    <h3 *ngFor="let param of parametro" class="ion-text-center ">
      <b>{{param['date'] |slice:0:10 }}</b>
    </h3>
  </ion-text>
<!-- TEMPERATURA -->
  <ion-text color="dark" class="custom-font ">
    <h2> <ion-icon  name="thermometer-outline"></ion-icon> Temperatura (°C)</h2><br>
  </ion-text>
  <div class="scroll-container ">
    <ngx-charts-bar-vertical
  *ngIf="data"
  [view]="[475, 100]"
  [scheme]="barChartOptions.colorScheme"
  [results]="data[0].series"
  [gradient]="barChartOptions.gradient"
  [xAxis]="barChartOptions.showXAxis"
  [yAxis]="barChartOptions.showYAxis"
  [showXAxisLabel]="barChartOptions.showXAxisLabel"
  [showYAxisLabel]="barChartOptions.showYAxisLabel"
  [xAxisLabel]="barChartOptions.xAxisLabel"
  [yAxisLabel]="barChartOptions.yAxisLabel"
  [showGridLines]="barChartOptions.showGridLines"
  [barPadding]="barChartOptions.barPadding"
  [yAxisTickFormatting]="yAxisTickFormatting1"
  [xAxisTickFormatting]="xAxisTickFormatting"
  >
</ngx-charts-bar-vertical>
</div>
<!-- HUMEDAD SUELO -->
 <ion-text color="primary">
```

Figura 39: Código página de reportes

3.8.4 Prueba

Para cumplimiento del tercer objetivo de investigación que indica la evaluación de la fiabilidad de la aplicación móvil para el monitoreo de cultivo de tomate, se evaluó mediante la herramienta Apache JMeter. Se realizaron 250 solicitudes con diferentes peticiones por segundo. Después, la información arrojada por la herramienta fue analizada, y se representó

a través de gráficos estadísticos. Este análisis se comparó con los estándares de calidad del modelo FURPS para cada indicador.

a) Pruebas del sistema y de integración.

Se requiere del análisis de los indicadores propuestos para evaluar la fiabilidad de la aplicación móvil para el monitoreo de cultivos. A continuación, en la tabla 12 se muestran los indicadores clave a evaluar para la realización de las pruebas.

Tabla 12: Parámetros de evaluación

Medida	Indicadores
Fiabilidad	% de recuperabilidad. % de precisión. % predicción

- **Ejecución de pruebas**

En la figura 40, se muestra la configuración inicial de la herramienta para realizar las peticiones, delineando los parámetros esenciales para cada requerimiento.

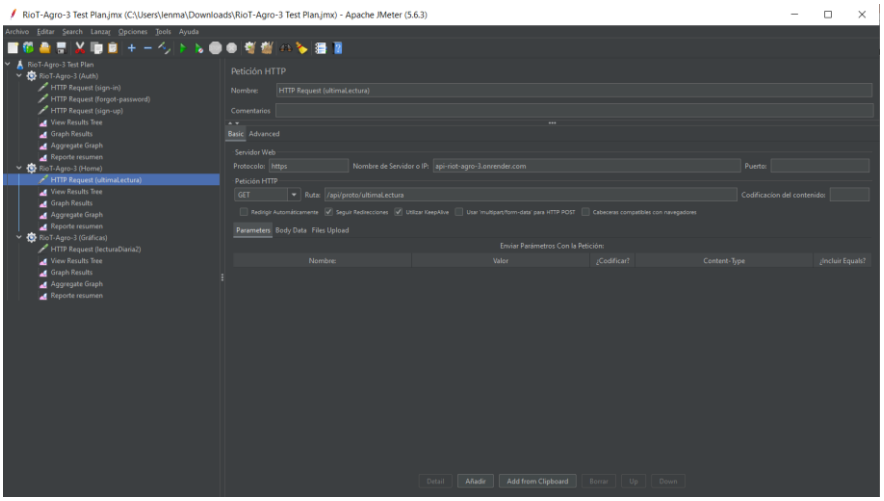


Figura 40: Configuración del escenario JMete

A continuación, se muestra la configuración de los hilos para cada prueba en donde se coloca el tiempo de ejecución de las pruebas en la herramienta JMeter como se muestra en la Figura 41.

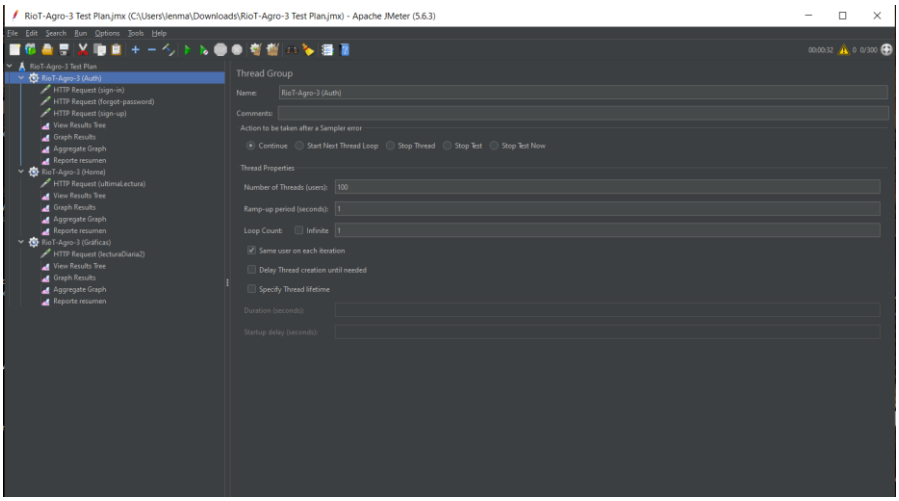


Figura 41: Configuración de los hilos del escenario

En la figura 42 se muestran los resultados iniciales de las pruebas ejecutadas. Los datos resultantes muestran a detalle una interpretación básica de la aplicación bajo las diferentes pruebas.

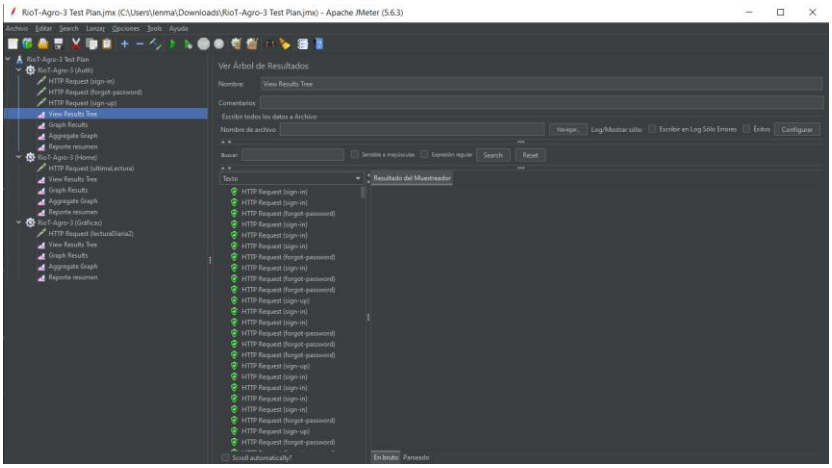


Figura 42: Resultados generales (árbol de resultados)

También se generó una gráfica de los resultados de los módulos mostrados por la herramienta para su posterior análisis.

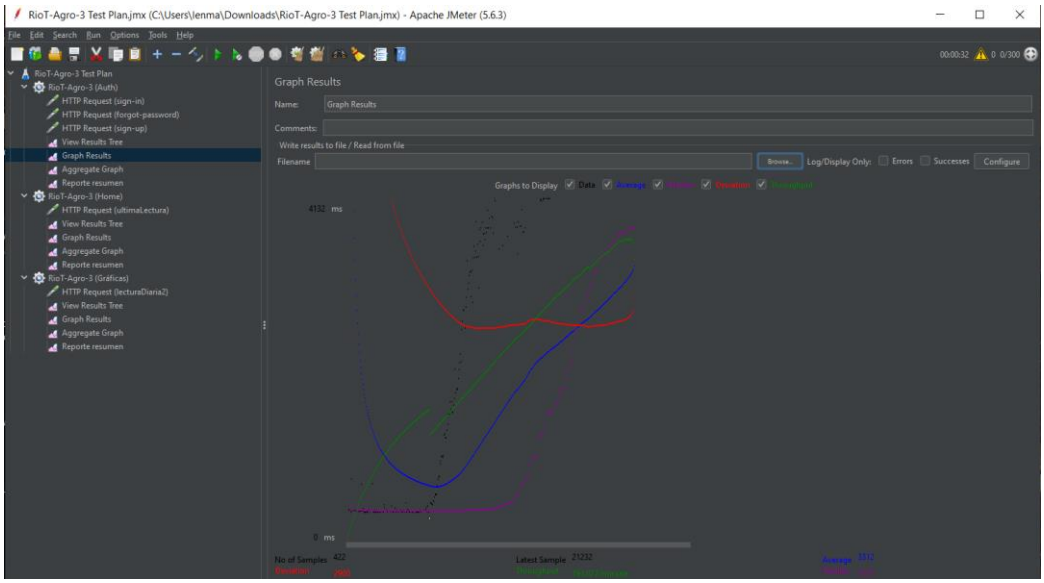


Figura 43: Gráfica de resultados

CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

4.1 Resultados

El desarrollo de la aplicación móvil y prototipo de monitoreo para cultivos de tomate en huertos urbanos se ha realizado exitosamente. Desarrollada mediante tecnologías como Ionic (Angular, HTML, CSS, Typescript), Express.js y MongoDB para la aplicación móvil y tecnologías de microcontroladores, sensores, diseño 3D de producción y conocimientos de programación dentro del entorno Arduino IDE en la parte del dispositivo electrónico de IoT. La aplicación móvil y su respectivo prototipo de monitoreo ofrecen un entorno amigable para el monitoreo de la información de los parámetros ambientales de los cultivos.
En la figura 44 se observan las interfaces principales de la aplicación móvil de monitoreo.

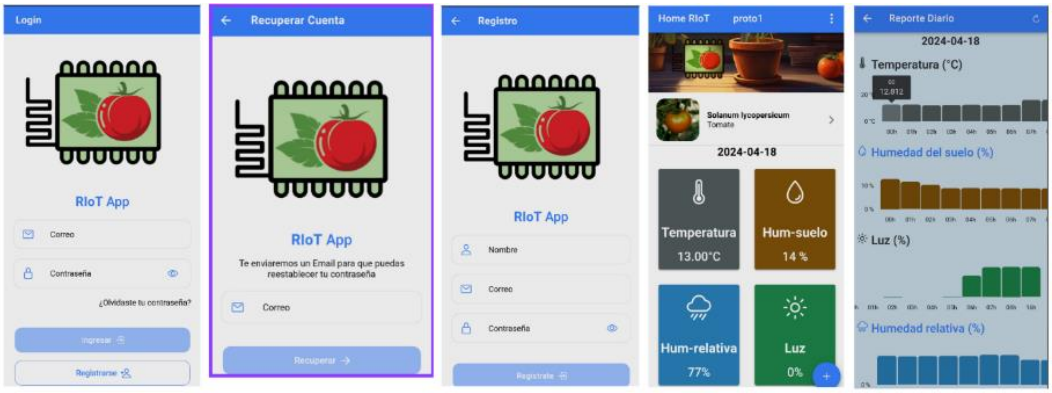


Figura 44: Interfaces principales de la aplicación móvil RIoT

Resultados de la Evaluación de fiabilidad de la aplicación móvil

Una vez concluido el desarrollo del prototipo de monitoreo para cultivos de tomate en huertos urbanos mediante una aplicación móvil, se realizaron las pruebas de fiabilidad para posteriormente interpretar sus resultados a continuación. Se muestra el análisis de las pruebas realizadas de la aplicación móvil de monitoreo. Se utiliza la herramienta para evaluar aplicaciones Apache JMeter.

4.1.1 Valoración de los indicadores

Para valorar los indicadores se presentan gráficos que van de acuerdo con cada indicador evaluado, lo que constituye un apartado específico dedicado a la evaluación detallada de estos parámetros. Cada grafico ofrece una representación visual de la fiabilidad de la aplicación móvil con referencia a los indicadores propuestos. Este modelo facilita la comprensión profunda y a detalle de la evaluación, ofreciendo información crucial sobre el sistema en cada aspecto considerado.

Recuperabilidad

Para evaluar la recuperabilidad de la aplicación se realizó un escenario en caso de que el servidor no envíe los datos de los parámetros ambientales recibidos por el prototipo de medición a la aplicación móvil. Dando como resultado que la aplicación muestra los campos en blanco hasta que se reestablezca una conexión con el servidor. Por otro lado, en el caso de que el dispositivo de monitoreo pierda la conexión con la base de datos la aplicación mostró los últimos datos que se subieron a la base de datos sin dejar de funcionar.

La tabla 13 y la figura 45 a continuación, se muestra el porcentaje de recuperabilidad de la aplicación móvil en relación con los 300 procesos de simulación de desconexión con el servidor de base de datos ejecutados. El resultado del indicador demuestra que en todos los casos que se examinaron la recuperabilidad es del 100%.

Tabla 13: Resultados del indicador de Recuperabilidad

Parámetro	Indicador	Pruebas de desconexión con la base de datos
Recuperabilidad	Procesos realizados con éxito	300 procesos de desconexión con la base de datos
	Procesos realizados con error	0 procesos de desconexión con la base de datos
	Porcentaje	100%

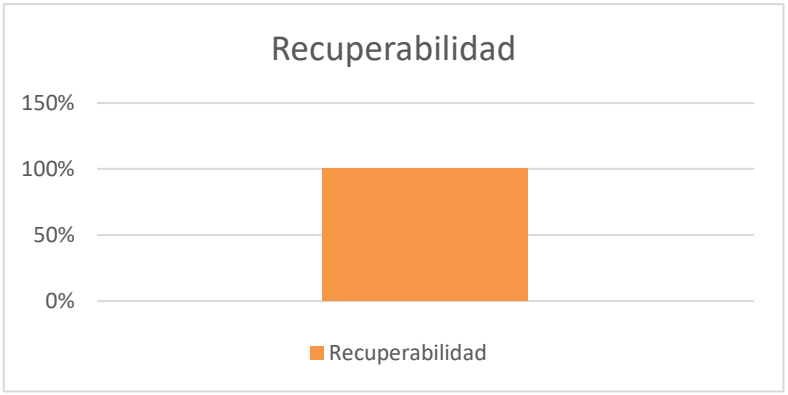


Figura 45: Indicador Recuperabilidad

Precisión

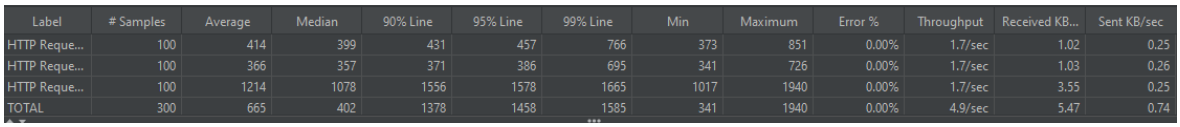
Para evaluar la precisión de las peticiones HTTP de la aplicación móvil se mide la capacidad de la aplicación móvil para enviar solicitudes correctamente al servidor y recibir respuestas precisas. Se identificaron las peticiones más importantes para el funcionamiento de la aplicación mediante sus correspondientes endpoints desarrollados.

En la tabla 15 y la figura 46, 47 y 48 se muestran los resultados de la precisión indicando que la aplicación realiza correctamente los procesos según lo esperado por el indicador dando como resultado el 100% de precisión luego de realizados los 300 procesos de peticiones en un segundo y 0% de error.

Tabla 14: Resultado de indicador de Precisión de procesos

Parámetro	Indicador	Peticiones realizadas
Precisión	Número de peticiones correctas realizadas a los módulos de autenticación, home y reportes diarios.	300 peticiones HTTP

	Número de peticiones incorrectas realizadas a los módulos de autenticación, home y reportes diarios.	0 peticiones HTTP
	Porcentaje	100%



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB...	Sent KB/sec
HTTP Reque...	100	414	399	431	457	766	373	851	0.00%	1.7/sec	1.02	0.25
HTTP Reque...	100	366	357	371	386	695	341	726	0.00%	1.7/sec	1.03	0.26
HTTP Reque...	100	1214	1078	1556	1578	1665	1017	1940	0.00%	1.7/sec	3.55	0.25
TOTAL	300	665	402	1378	1458	1585	341	1940	0.00%	4.9/sec	5.47	0.74

Figura 46: Resultados de precisión de cada módulo

En la figura 47 visualiza cómo se debe reflejar cada petición realizada con éxito a los endpoints recibiendo el mensaje en la pestaña "Sampler result" un código de 200 del servidor y una respuesta de OK. Además, en la pestaña de "Response data" se puede visualizar la información que ha llegado de la petición realizada.

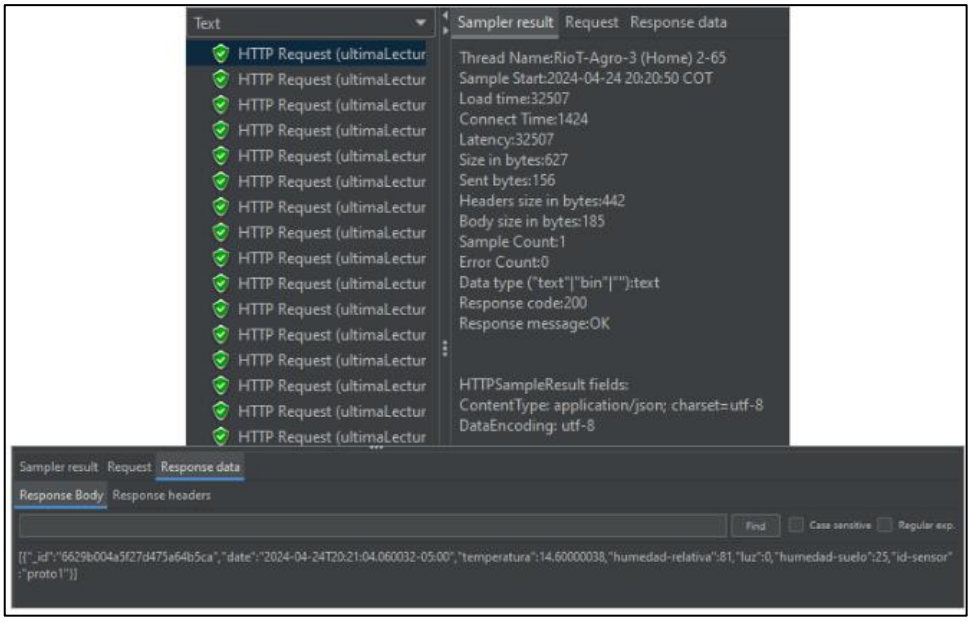


Figura 47: Mensaje de éxito para peticiones HTTP de los módulos

Predicción

Evaluar la capacidad de predicción en el apartado de confiabilidad del modelo de calidad FURPS ha implicado diversas pruebas y análisis para poder determinar si el sistema puede prever de manera precisa y confiable eventos futuros, como fallos o comportamientos inesperados. Para este caso se hicieron pruebas de estrés y carga realizados con JMeter mostrando un escenario de prueba que simuló condiciones extremas en el sistema para observar cómo responde la aplicación y evaluar su capacidad para predecir y manejar situaciones de muy alta demanda.

Para calcular el porcentaje de predicción de la evaluación se hizo el siguiente cálculo matemático.

$$Porcentaje\ de\ Predicción = \left(\frac{Número\ de\ solicitudes\ exitosas}{Numero\ total\ de\ solicitudes}\right) \times 100$$

La simulación de la carga se la realizo de manera que en 15 segundos se hayan conectado 1000 usuarios en un solo ciclo, la misma carga se colocó en el módulo de autenticación, home y reportes dando como total al finalizar la prueba 3000 solicitudes (véase figura 48). El resultado de la prueba indica que el número de solicitudes exitosas es 2864 de 3000 en total que se realizaron como se muestra en la figura 49. Usando la formula del porcentaje de predicción se obtiene un resultado del 95.46 %.

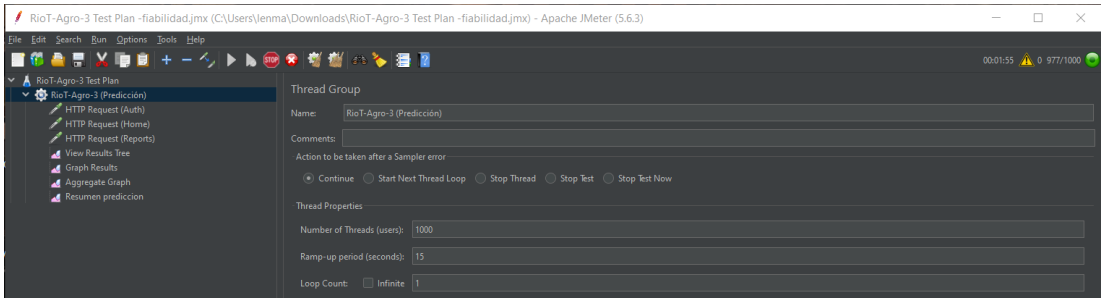


Figura 48: Prueba de carga para predicción

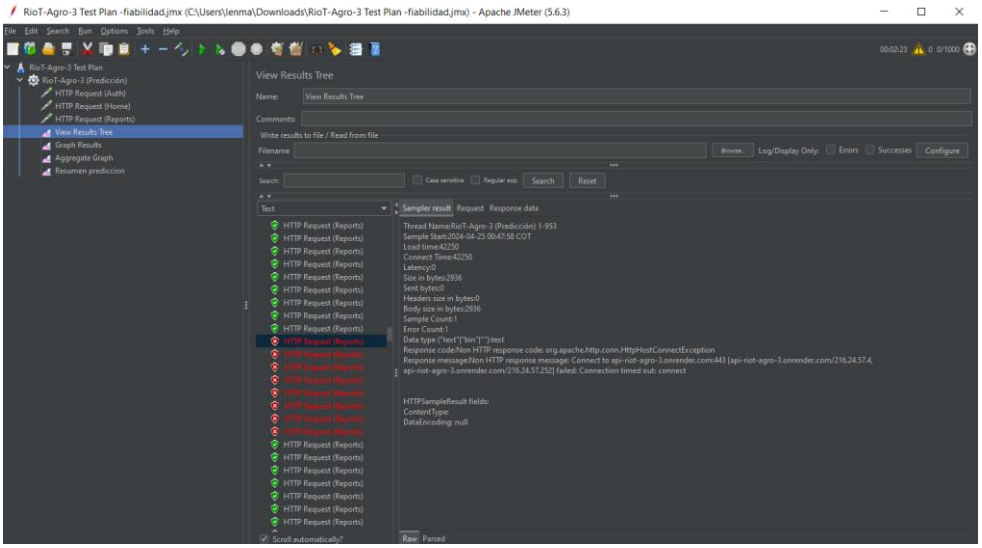


Figura 49: Resultado de la prueba de carga para la predicción.

4.2 Discusión

De acuerdo con Yeomans Daniel FURPS establece una puntuación en tres niveles, el primer nivel (alto) de cumplimiento va desde el 95% al 100%, en donde el software es altamente recomendado, el segundo nivel (moderado) desde el 80% al 95%, aquí el software deberá ser corregido en ciertos factores y el riesgo es controlable, el ultimo nivel (bajo) posee un cumplimiento menor al 80% por lo que el producto de software no cumple con los factores evaluados y el riesgo a ser utilizado es alto. (Yeomans, 2017)

Para medir los indicadores de fiabilidad en la aplicación móvil de monitoreo para cultivos de tomate en JMeter el porcentaje de recuperabilidad fue del cien por ciento, el porcentaje de precisión de la aplicación resulta tener un cien por ciento cuando la aplicación hace

solamente 300 peticiones por segundo y en el caso de la predicción de la aplicación indica un 87 %. Sí se hace un promedio del porcentaje de los 3 aspectos evaluados se obtiene un valor de 95.46% indicando que el resultado de la evaluación de la fiabilidad se encuentra dentro del rango desde el 95% al 100% según el modelo de calidad FURPS. Por lo tanto, se considera que la aplicación móvil es de calidad.

Existe una amplia lista de aplicaciones similares a las que se realizó en la presente investigación, la que se asimilas más es el proyecto realizado por (Rojas, 2021) de la Universidad Autónoma de Bucaramanga titulado "Sistema de monitoreo y control en tiempo real para un invernadero de tomate a través de un aplicativo móvil. Haciendo un comparativo entre ambos proyectos se puede notar que una de las aplicaciones está orientada a la implementación de un sistema de monitoreo y control en tiempo real de bajo costo para un invernadero de tomate usando tecnología de código abierto. Por otra parte, el "Prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic" se centra únicamente en el monitoreo de cultivos de tomate en huertos urbanos en general, además de, mostrar tablas de reportes diarios por hora almacenados en una base de datos de los parámetros ambientales y físicos que recolectan los sensores.

CAPÍTULO V. CONCLUSIONES y RECOMENDACIONES

5.1 Conclusiones

- Tras investigar aplicaciones móviles de IoT (Internet de las cosas) para monitoreo de cultivos se obtuvo información sobre el potencial que las aplicaciones pueden aportar al momento de obtener datos de parámetros ambientales, el estado de los cultivos brinda a los agricultores o usuarios herramientas poderosas y novedosas para mejorar la eficiencia, optimizar los recursos y aumentar la productividad.
- Se desarrolló la aplicación móvil mediante el framework Ionic que está construido sobre Angular, tecnologías como HTML, CSS y Typescript para el frontend, para el desarrollo del backend se usó MongoDB Atlas, el Framework Express.js para la parte del servidor encargado de las solicitudes HTTP de la aplicación. La implementación de la metodología de cascada o "Waterfall" fue fundamental para el desarrollo progresivo del aplicativo móvil. Esta metodología proporciono una estructura secuencial que permitió avanzar de manera ordenada desde la etapa de análisis hasta sus pruebas proporcionando un marco eficiente para el desarrollo del sistema.
- La evaluación de la fiabilidad de la aplicación móvil utilizando el modelo de calidad FURPS ha demostrado que cumple con los criterios establecidos, ofreciendo resultados favorables en sus indicadores que muestran un 100% de recuperabilidad, un 100% de precisión y un 87% en el indicador de predicción dando como resultado que la aplicación es de calidad.

5.2 Recomendaciones

- Se recomienda usar herramientas de programación adecuadas para que la codificación de aplicaciones de internet de las cosas siempre arroje datos fiables y se pueda mantener los cultivos de los huertos urbanos en estado óptimo. En los sensores utilizados se debe enlazar correctamente con la base de datos para que la información obtenida mediante la aplicación sea real.
- Al crear un producto final debe tener una buena estructura y el uso de herramientas adecuadas en su desarrollo para que no existan posibles complicaciones o perdida de datos al momento de su uso a futuro. Esto es muy importante dado que la herramienta debe estar siempre en buenas condiciones para que los datos sean fiables y se obtenga un buen monitoreo en los cultivos.
- Dado que la evaluación ha demostrado que la aplicación móvil cumple con los criterios de fiabilidad y ha mostrado una calidad satisfactoria incluso en condiciones adversas, sería prudente recomendar su posible implementación piloto en un entorno agrícola real. Se recomienda llevar un monitoreo exhaustivo del sistema para analizar que los datos a futuro no arrojen información errónea ya sea por degradación de software o complicaciones externas.

BIBLIOGRAFÍA

Agüero, J. M. (2021). *profile.es*. Obtenido de https://profile.es/blog/que-es-ionic/

Ana, D. (2023). *hostinguer*. Obtenido de https://www.hostinger.es/tutoriales/que-es-angular

Andrade, J. M. (2022). *criptonube*. Obtenido de https://criptonube.com/2022/11/firebase/

App&Web. (2019). Obtenido de https://www.appandweb.es/blog/iot-agricultura/

azure microsoft. (2023). Obtenido de https://azure.microsoft.com/es-es/solutions/iot/iot-technology-protocols/

Banco Mundial. (2022). Obtenido de https://www.bancomundial.org/es/topic/urbandevelopment/overview#:~:text=Hoy%20en%20d%C3%ADa%2C%20alrededor%20del,10%20personas%20vivir%C3%A1n%20en%20ciudades.

barbaraiot. (2021). Obtenido de https://barbaraiot.com/es/blog/protocolos-iot-que-deberias-conocer#:~:text=En%20el%20ámbito%20de%20la,facilitando%20la%20comunicaciónMachine2Machine%20(M2M).

Bassi, A. (2021). *Goto IoT*. Obtenido de https://www.gotoiot.com/pages/articles/iot_protocols_intro/index.html

Carmenate, J. G. (2021). *programarfacil*. Obtenido de https://programarfacil.com/blog/arduino-blog/arduino-ide/

Consulting Informático. (2021). Obtenido de https://www.cic.es/iot-sus-aplicaciones/#:~:text=El%20Internet%20de%20las%20Cosas%20(IoT)%20hace%20referencia%20a%20una,la%20sanidad%2C%20la%20administraci%C3%B3n%20p%C3%BAblica%E2%80%A6

Coppola, M. (2022). *hubspot*. Obtenido de https://blog.hubspot.es/website/que-es-angular

Degenhart, B. (2016). *nuso.org*. Obtenido de https://nuso.org/articulo/la-agricultura-urbana-un-fenomeno-global/

educo. (2019). Obtenido de https://www.educo.org/blog/las-ventajas-de-tener-un-huerto-urbano-en-casa

flexbot. (2020). Obtenido de https://www.flexbot.es/estructura-basica-del-codigo-arduino/

García, G. (2022). *thefoodtech*. Obtenido de https://thefoodtech.com/seguridad-alimentaria/esta-es-la-contribucion-de-la-tecnologia-a-la-agricultura-inteligente/

Gonzalves, M. J. (2021). *Hiberus*. Obtenido de https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/

google Cloud. (2020). Obtenido de https://cloud.google.com/mongodb?hl=es-419#:~:text=MongoDB%20Atlas%20en%20Google%20Cloud,administrativas%20que%20llevan%20mucho%20tiempo.

hidroenv. (2018). Obtenido de https://www.hydroenv.com.mx/catalogo/index.php?main_page=page&id=384

IAT . (2020). Obtenido de https://iat.es/tecnologias/internet-de-las-cosas-iot/agricultura/

innovacion&tecnologia. (2020). Obtenido de https://www.innovacion-tecnologia.com/iot/plataformas-iot/

Lanner. (2016). Obtenido de https://www.lanner-america.com/es/iot/

Lavín, I. (2022). *elmueble*. Obtenido de https://www.elmueble.com/plantas-flores/huerto-urbano_48321

Nuñez, J. (2017). *costaricamakers.com*. Obtenido de https://costaricamakers.com/iot-para-hogares-inteligentes/

oasys. (2022). Obtenido de https://oasys-sw.com/que-es-iot-platform-industria/#:~:text=Una%20plataforma%20IoT%20logra%20conectar,conexi%C3%B3n%20directa%20o%20en%20pasarela).

RedHat. (2019). Obtenido de https://www.redhat.com/es/topics/internet-of-things/what-is-iot#:~:text=El%20Internet%20de%20las%20cosas%20(IoT)%20es%20el%20proceso%20que,accesorios%20personales%20inteligentes%3B%20e%20incluso

Royce, W. (2019). *Ionos*. Obtenido de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/

Stsepanets, A. (2023). *Ganttpro*. Obtenido de https://blog.ganttpro.com/es/metodologia-de-cascada/#:~:text=La%20primera%20descripción%20formal%20de,la%20fabricación%20y%20la%20construcción.

Tokioschool. (2022). Obtenido de https://www.tokioschool.com/noticias/sensores-iot/#:~:text=Los%20sensores%20IoT%20o%20sensores,de%20Internet%20de%20las%20cosas.

Unit Electronics. (2018). Obtenido de https://uelectronics.com/producto/sensor-de-temperatura-y-humedad-dht22-con-cables/#:~:text=Módulo%20DHT22%20es%20un%20sensor,en%20el%20pin%20de%20datos.

ANEXOS

Anexo 1: Código server api

```
var Express = require("express");
var MongoClient= require("mongodb").MongoClient;
var cors=require("cors");
const multer=require("multer");
const { error } = require("console");

var app=Express();
app.use(cors());

var
CONNECTION_STRING="mongodb+srv://lenmanri29:JKnDqGdq8FYR7Zhg@cluster0.snrto5
3.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0";
const PORT = process.env.PORT || 5038;
var DATABASENAME="prototipo";
var database;
app.listen(5038,()=>{
    MongoClient.connect(CONNECTION_STRING,(error,client)=>{
        database=client.db(DATABASENAME);
        console.log("Mongo DB conexion exitosa...");
    });

})
app.get('/api/proto/parametros',(request,response)=>{
        database.collection("parametros-
planta").find({}).toArray((error,result)=>{
        response.send(result);
    });
})
//obetner el ultimo documento
app.get('/api/proto/ultimaLectura', (request, response) => {
    database.collection("parametros-planta")
    .find({})
    .sort({_id: -1})
    .limit(1)
    .toArray((error,result)=>{
        if (error) throw error;
        response.send(result);
    });
})
//lecturas por fecha//
app.get('/api/proto/lecturaDiaria', (request, response) => {
    // Obtener la fecha del último documento devuelto por el primer endpoint
    database.collection("parametros-planta")
    .find({})
```

```
    .sort({_id: -1})
    .limit(1)
    .toArray((error,result)=>{
        if (error) throw error;
        // Extraer los primeros 10 caracteres de la fecha del primer
documento devuelto
        const fechaDocumento = result[0].date.substring(0, 10);
        // Consultar la colección para obtener todos los documentos que
coincidan con la fecha
        database.collection("parametros-planta")
        .find({
            "date": { $regex: new RegExp("^" + fechaDocumento) } // Utilizar
expresión regular para comparar los primeros 10 caracteres
        })
        .toArray((error, resultados) => {
            if (error) throw error;
            response.send(resultados);
        });
    });
});
//===== Lectura diaria promedio hora====//
app.get('/api/proto/lecturaDiariaPromedio', (request, response) => {
    database.collection("parametros-planta")
    .find({})
    .sort({_id: -1})
    .limit(1)
    .toArray((error, result) => {
        if (error) throw error;
        const fechaDocumento = result[0].date.substring(0, 10);
        // Consultar la colección para obtener todos los documentos que
coincidan con la fecha
        database.collection("parametros-planta")
        .find({
            "date": { $regex: new RegExp("^" + fechaDocumento) }
        })
        .toArray((error, resultados) => {
            if (error) throw error;
            // Objeto para almacenar los promedios por hora
            const promediosPorHora = {};
            // Iterar sobre los documentos y calcular los promedios
            resultados.forEach(documento => {
                const hora = documento.date.substring(11, 13); // Obtener la
hora del documento
                if (!promediosPorHora[hora]) {
                    // Inicializar el objeto para esta hora si no existe
                    promediosPorHora[hora] =
                     {
                        "hora": parseInt(hora),
```

```
                        temperatura: [],
                        "humedad-relativa": [],
                        luz: [],
                        "humedad-suelo": []
                    };
                }
                // Agregar los valores a los arrays correspondientes
                promediosPorHora[hora].temperatura.push(documento.temperatur
a);
                promediosPorHora[hora]["humedad-
relativa"].push(documento["humedad-relativa"]);
                promediosPorHora[hora].luz.push(documento.luz);
                promediosPorHora[hora]["humedad-
suelo"].push(documento["humedad-suelo"]);
            });
            // Calcular los promedios por hora
            const promedios = {};
            for (const hora in promediosPorHora) {
                const valores = promediosPorHora[hora];
                promedios[hora] = {
                    "hora":hora,
                    temperatura: promedio(valores.temperatura),
                    "humedad-relativa": promedio(valores["humedad-
relativa"]),
                    luz: promedio(valores.luz),
                    "humedad-suelo": promedio(valores["humedad-suelo"])
                };
            }
            const resultadoFinal = Object.values(promedios);
            response.send(resultadoFinal);
        });
    });
});
// Función para calcular el promedio de un array de números
function promedio(array) {
    const sum = array.reduce((acc, val) => acc + val, 0);
    return sum / array.length;
}
//=====endpoint lescturas por dia promedio======//
app.get('/api/proto/lecturaDiaria2', (request, response) => {
    database.collection("parametros-planta")
    .find({})
    .sort({_id: -1})
    .limit(1)
    .toArray((error, result) => {
        if (error) throw error;
        const fechaDocumento = result[0].date.substring(0, 10);
```

```
        // Consultar la colección para obtener todos los documentos que
coincidan con la fecha
        database.collection("parametros-planta")
        .find({
            "date": { $regex: new RegExp("^" + fechaDocumento) }
        })
        .toArray((error, resultados) => {
            if (error) throw error;
            // Objeto para almacenar los promedios por hora
            const promediosPorHora = {};
            // Iterar sobre los documentos y calcular los promedios
            resultados.forEach(documento => {
                let hora = documento.date.substring(11, 13); // Obtener la
hora del documento
                if (!promediosPorHora[hora]) {
                    // Inicializar el objeto para esta hora si no existe
                    promediosPorHora[hora] =
                    {
                        "hora": parseInt(hora),
                        temperatura: [],
                        "humedad-relativa": [],
                        luz: [],
                        "humedad-suelo": []
                    };
                }
                // Agregar los valores a los arrays correspondientes
                promediosPorHora[hora].temperatura.push(documento.temperatur
a);
                promediosPorHora[hora]["humedad-
relativa"].push(documento["humedad-relativa"]);
                promediosPorHora[hora].luz.push(documento.luz);
                promediosPorHora[hora]["humedad-
suelo"].push(documento["humedad-suelo"]);
            });
            // Calcular los promedios por hora
            const promedios = {};
            for (const hora in promediosPorHora) {
                const valores = promediosPorHora[hora];
                promedios[hora] = {
                    "hora":hora,
                    temperatura: promedio(valores.temperatura),
                    "humedad-relativa": promedio(valores["humedad-
relativa"]),
                    luz: promedio(valores.luz),
                    "humedad-suelo": promedio(valores["humedad-suelo"])
                };
            }
            const resultadoFinal = Object.values(promedios);
```

```
            // Ordenar el array de objetos en función del campo "hora"
            resultadoFinal.sort((a, b) => {
                // Convertir las horas a números antes de comparar
                const horaA = parseInt(a.hora);
                const horaB = parseInt(b.hora);
                return horaA - horaB;
            });
            response.send(resultadoFinal);
        });
    });
});
// Función para calcular el promedio de un array de números
function promedio(array) {
    const sum = array.reduce((acc, val) => acc + val, 0);
    return sum / array.length;

}
```

Anexo 2: Código home page HTML

```
<ion-header>
  <ion-toolbar *ngFor="let param of parametro" color="primary">
      <ion-title slot="start">
        Home RIoT
               
        {{param['id-sensor']}}
      </ion-title>
      <ion-buttons slot="end" >
        <ion-button (click)="openMenu()">
          <ion-icon slot="icon-only" name="ellipsis-vertical"></ion-icon>
        </ion-button>
      </ion-buttons>
  </ion-toolbar>
</ion-header>
<!-- =============== contenido ================ -->
<ion-content >
<div class="video-container ">
  <video autoplay loop muted class="video">
    <source src="assets/videoHome.mp4" type="video/mp4" >
  </video>
</div>
<!-- ================== Info Tomate============== -->
<div>
  <ion-item lines="full" class="larger-item"  routerLink="../info-tomate" >

    <ion-avatar slot="start" class="avatar-large">
      <img alt="Solanum lycopersicum"
src="assets/pictures/tomatePerfil1.jpg" />
```

```
    </ion-avatar>
    <ion-label><b>Solanum lycopersicum </b> <br> Tomate</ion-label>
    <ion-icon slot="end" name="chevron-forward"></ion-icon>
  </ion-item>
</div>
<!-- <p *ngFor="let param of parametro">{{ formatDate(param.date) }}</p> -->
<h3 *ngFor="let param of parametro" class="ion-text-center ion-no-margin">
  <b>{{param['date'] |slice:0:10 }}</b>
</h3>
<ion-grid fixed >
  <ion-row>
    <!-- ==========TEMPERATURA========= -->
    <ion-col size="6">
      <ion-card color="medium"  class="card-temperatura">
        <ion-icon class="icono-grande" name="thermometer-outline"></ion-
icon>
         <ion-text  >
          <h1><b>Temperatura</b></h1>
          <h2 *ngFor="let param of parametro" >
            {{param.temperatura | number:'1.2-2'}}°C
          </h2>
         </ion-text>
      </ion-card>
    </ion-col>
    <!-- ===========HUMEDAD SUELO=========== -->
    <ion-col size="6">
      <ion-card color="success" class="card-Humedad-suelo" >
        <ion-icon class="icono-grande" name="water-outline"></ion-icon>
        <ion-text >
          <h1><b>Hum-suelo</b></h1>
          <h2 *ngFor="let param of parametro" >{{param['humedad-suelo']}}
%</h2>
         </ion-text>
      </ion-card>
    </ion-col>
    <!-- ===========HUMEDAD RELATIVA========== -->
    <ion-col size="6" >
      <ion-card color="success" class="card-Humedad-relativa" >
        <ion-icon class="icono-grande" name="rainy-outline"></ion-icon>
        <ion-text >
          <h1 ><b>Hum-relativa</b></h1>
          <h2 *ngFor="let param of parametro">{{param['humedad-
relativa']}}%</h2>
         </ion-text>
      </ion-card>
    </ion-col>
    <!-- ============== LUZ ================ -->
    <ion-col size="6">
```

```
      <ion-card color="success" class="card-luz">
        <ion-icon class="icono-grande" name="sunny-outline"></ion-icon>
        <ion-text  >
          <h1><b>Luz</b></h1>
          <h2 *ngFor="let param of parametro">{{param['luz']}}%</h2>
         </ion-text>
      </ion-card>
    </ion-col>
  </ion-row>
</ion-grid>
  <!-- ========Nuevo Producto========= -->
  <ion-fab vertical="bottom" horizontal="end" slot="fixed">
    <ion-fab-button>
      <ion-icon name="add"></ion-icon>
    </ion-fab-button>
    <ion-fab-list side="top">
      <ion-fab-button><ion-icon name="extension-puzzle-outline"
color="danger"></ion-icon></ion-fab-button>
      <ion-fab-button><ion-icon name="bar-chart-outline" color="dark"
routerLink="../reports"></ion-icon></ion-fab-button>
      <!-- <ion-fab-button><ion-icon name="hardware-chip-outline"
color="dark" routerLink="../devices"></ion-icon></ion-fab-button> -->
    </ion-fab-list>
  </ion-fab>
</ion-content>
```

Anexo3: Código página reportes HTML.

```
<ion-header>
  <ion-toolbar color="primary">
    <ion-buttons slot="start">
      <ion-back-button defaultHref="/"></ion-back-button>
    </ion-buttons>
    <ion-title>Reporte Diario</ion-title>
    <ion-buttons slot="end">
      <ion-button>
        <ion-icon slot="" name="refresh"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<!-- ====== reportes ========== -->
<ion-content>
  <ion-text color="dark" class="ion-text-center">
    <h3 *ngFor="let param of parametro" class="ion-text-center ">
      <b>{{param['date'] |slice:0:10 }}</b>
```

```
    </h3>
  </ion-text>
<!-- TEMPERATURA -->
  <ion-text color="dark" class="custom-font ">
    <h2> <ion-icon  name="thermometer-outline"></ion-
icon> Temperatura (°C)</h2><br>
  </ion-text>
  <div class="scroll-container ">
    <ngx-charts-bar-vertical
  *ngIf="data"
  [view]="[475, 100]"
  [scheme]="barChartOptions.colorScheme"
  [results]="data[0].series"
  [gradient]="barChartOptions.gradient"
  [xAxis]="barChartOptions.showXAxis"
  [yAxis]="barChartOptions.showYAxis"
  [showXAxisLabel]="barChartOptions.showXAxisLabel"
  [showYAxisLabel]="barChartOptions.showYAxisLabel"
  [xAxisLabel]="barChartOptions.xAxisLabel"
  [yAxisLabel]="barChartOptions.yAxisLabel"
  [showGridLines]="barChartOptions.showGridLines"
  [barPadding]="barChartOptions.barPadding"
  [yAxisTickFormatting]="yAxisTickFormatting1"
  [xAxisTickFormatting]="xAxisTickFormatting"
  >
</ngx-charts-bar-vertical>
</div>
<!-- HUMEDAD SUELO -->
 <ion-text color="primary">
    <h2> <ion-icon  name="water-outline"></ion-icon> Humedad del
suelo (%)</h2><br>
  </ion-text>
<div class="scroll-container">
  <ngx-charts-bar-vertical
  *ngIf="data"
  [view]="[475, 100]"
  [scheme]="barChartOptions2.colorScheme"
  [results]="data[3].series"
  [gradient]="barChartOptions.gradient"
  [xAxis]="barChartOptions.showXAxis"
  [yAxis]="barChartOptions.showYAxis"
  [showXAxisLabel]="barChartOptions.showXAxisLabel"
  [showYAxisLabel]="barChartOptions.showYAxisLabel"
  [xAxisLabel]="barChartOptions.xAxisLabel"
  [yAxisLabel]="barChartOptions.yAxisLabel"
  [showGridLines]="barChartOptions.showGridLines"
  [barPadding]="barChartOptions.barPadding"
  [yAxisTickFormatting]="yAxisTickFormatting2"
```

```
  [xAxisTickFormatting]="xAxisTickFormatting">
</ngx-charts-bar-vertical>
</div>
<!-- luz  -->
<ion-text color="dark">
  <h2> <ion-icon  name="sunny-outline"></ion-icon> Luz
(%)</h2><br>
</ion-text>
<div class="scroll-container">
  <ngx-charts-bar-vertical
  *ngIf="data"
  [view]="[475, 100]"
  [scheme]="barChartOptions3.colorScheme"
  [results]="data[2].series"
  [gradient]="barChartOptions.gradient"
  [xAxis]="barChartOptions.showXAxis"
  [yAxis]="barChartOptions.showYAxis"
  [showXAxisLabel]="barChartOptions.showXAxisLabel"
  [showYAxisLabel]="barChartOptions.showYAxisLabel"
  [xAxisLabel]="barChartOptions.xAxisLabel"
  [yAxisLabel]="barChartOptions.yAxisLabel"
  [showGridLines]="barChartOptions.showGridLines"
  [barPadding]="barChartOptions.barPadding"
  [yAxisTickFormatting]="yAxisTickFormatting2"
  [xAxisTickFormatting]="xAxisTickFormatting">
</ngx-charts-bar-vertical>
</div>
<!-- humedad relativa -->
<ion-text color="primary">
  <h2> <ion-icon  name="rainy-outline"></ion-icon> Humedad
relativa (%)</h2><br>
</ion-text>
<div class="scroll-container">
  <ngx-charts-bar-vertical
  *ngIf="data"
  [view]="[475, 100]"
  [scheme]="barChartOptions4.colorScheme"
  [results]="data[1].series"
  [gradient]="barChartOptions.gradient"
  [xAxis]="barChartOptions.showXAxis"
  [yAxis]="barChartOptions.showYAxis"
  [showXAxisLabel]="barChartOptions.showXAxisLabel"
  [showYAxisLabel]="barChartOptions.showYAxisLabel"
  [xAxisLabel]="barChartOptions.xAxisLabel"
  [yAxisLabel]="barChartOptions.yAxisLabel"
  [showGridLines]="barChartOptions.showGridLines"
  [barPadding]="barChartOptions.barPadding"
  [yAxisTickFormatting]="yAxisTickFormatting2"
```

```
  [xAxisTickFormatting]="xAxisTickFormatting">
</ngx-charts-bar-vertical>
</div>
</ion-content>
```

Anexo 4: Manual de usuario.

FACULTAD DE
Ingeniería

UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano mediante una aplicación móvil usando Ionic

RIoT

MANUAL DE USUARIO

Periodo académico 2023 - 2s

ÍNDICE

Contenido

1. INTRODUCCIÓN

Bienvenido al manual de usuario del prototipo de IoT para el monitoreo del cultivo de tomate en un huerto urbano. Este documento proporcionará una guía detallada sobre cómo utilizar el sistema **RIoT** para monitorear y gestionar el cultivo de tomates de manera eficiente.

2. OBJETIVO

El objetivo de este documento es guiar al usuario en el uso adecuado del prototipo IoT de monitoreo y su aplicación móvil.

3. FUNCIONES DE LA APP MÓVIL

- Log In (ingresar a la aplicación)
- Registrarse en la aplicación móvil
- Recuperar contraseña
- Inicio
- Menú
- Información
- Acerca de
- Reporte diario
- Dispositivo electrónico

4. CONTENIDO

4.1. Log In (ingresar a la aplicación móvil)

Interfaz principal para inicio de sesión del usuario.

4.2. Registrarse en la aplicación móvil

Seleccionamos el botón registrase de la interfaz Log In.

Nota: Tener en cuenta que el correo electrónico debe ser un correo válido.

4.3. Recuperar contraseña

Para recuperar la contraseña se debe seleccionar el texto "¿Olvidaste tu contraseña?" de la interfaz Log In.

Posterior a presionar el botón "Recuperar" se debe verificar la bandeja de entrada del correo electrónico proporcionado donde habrá llegado un enlace para reestablecer la contraseña.

El enlace va a redirigir a una interfaz para el cambio de contraseña.

4.4. Inicio

Después de haber iniciado sesión se mostrará la interfaz principal. Los parámetros que se muestran en tiempo real se refrescan automáticamente cada 20 segundos.

Al seleccionar la cruz ubicada en la parte inferior derecha se desplegará la opción de reportes diarios.

4.5. Menú

Al seleccionar la opción de menú en la página principal de despliega un menú que contiene lo siguiente.

Nota: La sesión se mantendrá abierta en el dispositivo siempre y cuando no se seleccione cerrar sesión para salir de la cuenta.

4.6. Acerca de

Cuando se selecciona la opción "acerca de" en la sección de menú se desplegará una página de información acerca del dispositivo y su desarrollador.

4.7. Reporte diario

En la página de inicio de la aplicación móvil al seleccionar la cruz de la parte inferior derecha se desplegará una opción que al seleccionarla nos va a redireccionar a la página de "reporte diario".

Las gráficas de tablas muestran los promedios por hora de las lecturas realizadas por el dispositivo. La actualización de las tablas es automática cada hora.

Dispositivo electrónico

El dispositivo al ser un prototipo puede ser reprogramado para alguna posible mejora de firmware. Se requiere el uso de un cable de datos USB-C y el software correspondiente (Arduino IDE o la extensión de Visual Studio Code llamada Platform.io)