Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Hito 1 Seminario y Proyecto de Titulo

“Proyecto Abejas”

Sebastian Farias Jiles – 20.012.499-5

INSW421 | PROYECTO DE TÍTULO

Contenido

[I. Introducción 2](#_Toc195228195)

[II. Fundamentación del Problema 3](#_Toc195228196)

[III. Contexto 5](#_Toc195228197)

[IV. Situación actual 5](#_Toc195228198)

[V. Técnica identificación de problemática 5](#_Toc195228199)

[VI. Descripción de causas de la problemática 6](#_Toc195228200)

[A) Hombre 6](#_Toc195228201)

[B) Máquina 7](#_Toc195228202)

[C) Entorno 7](#_Toc195228203)

[D) Material 7](#_Toc195228204)

[E) Método 7](#_Toc195228205)

[F) Medio Ambiente 7](#_Toc195228206)

[VII. Alcance del proyecto 7](#_Toc195228207)

[El proyecto considera: 8](#_Toc195228208)

[Fuera del alcance: 8](#_Toc195228209)

[VIII. Objetivo general 8](#_Toc195228210)

[IX. Objetivos específicos 8](#_Toc195228211)

[X. Métricas de objetivos 9](#_Toc195228212)

[XI. Alternativas de solución 9](#_Toc195228213)

[Mediciones manuales 9](#_Toc195228214)

[Ventajas 1](#_Toc195228215)

[Desventajas 1](#_Toc195228216)

[Sistema cableado 1](#_Toc195228217)

[Ventajas 1](#_Toc195228218)

[Desventajas 1](#_Toc195228219)

[Sistema de monitoreo remoto (solución propuesta) 1](#_Toc195228220)

[Ventajas 1](#_Toc195228221)

[Desventajas 1](#_Toc195228222)

[XII. Alcances y limitaciones del proyecto 1](#_Toc195228223)

[XIII. Factibilidades 2](#_Toc195228224)

[Factibilidad técnica 2](#_Toc195228225)

[Factibilidad económica 3](#_Toc195228226)

[Factibilidad Operativa 3](#_Toc195228227)

[XIV. Propuesta de solución 3](#_Toc195228228)

[XV. Diagrama de contexto 5](#_Toc195228229)

[XVI. Plan de proyecto 5](#_Toc195228230)

[XVIII. Metodología de desarrollo 6](#_Toc195228231)

[XIX. Metodología de desarrollo 6](#_Toc195228232)

[XX. Plan de monitoreo y control 6](#_Toc195228233)

[XXI. Plan de gestión de riesgos 7](#_Toc195228234)

[XXII. Diseño de proyecto de alto nivel 8](#_Toc195228235)

[Duración total 8](#_Toc195228236)

[Fases y tiempos de entrega: 8](#_Toc195228237)

[Recursos necesarios: 8](#_Toc195228238)

[Hitos: 9](#_Toc195228239)

[Dependencias: 9](#_Toc195228240)

[Recursos adicionales: 9](#_Toc195228241)

[XXIII. Vista lógica 10](#_Toc195228242)

[XXIV. Vista física 11](#_Toc195228243)

[Dispositivo de prueba actual (D1 mini sin módulo de redes móviles) 11](#_Toc195228244)

[Dispositivo final esperado (ESP32 con módulo de red) 11](#_Toc195228245)

[XXV. Vista de despliegue 12](#_Toc195228246)

[XXVI. Vista de escenarios 12](#_Toc195228247)

[XXVII. Iteración 1 13](#_Toc195228248)

[XXVIII. Duración y meta del Sprint 14](#_Toc195228249)

[XXIX. Funcionalidades a realizar 14](#_Toc195228250)

[XXX. Historias de usuario a realizar 14](#_Toc195228251)

[XXXI. Definición Sprint backlog 15](#_Toc195228252)

[XXXII. Diseño detallado 17](#_Toc195228253)

[Programación del dispositivo de prueba 19](#_Toc195228254)

[Programacion de la aplicación 25](#_Toc195228255)

[Pantalla home\_screen.dart 25](#_Toc195228256)

[Pantalla detalles\_screen.dart 30](#_Toc195228257)

[Configuración de AWS para recibir datos de los sensores 33](#_Toc195228258)

[secrets.h 35](#_Toc195228259)

[Creación de Rules para comunicación con Lambda 37](#_Toc195228260)

[Función Lambda 37](#_Toc195228261)

[Crear DynamoDB table 38](#_Toc195228262)

[Crear un Role para lambda a DynamoDB 39](#_Toc195228263)

[XXXIII. Riesgos gatillados 41](#_Toc195228264)

[XXXIV. Ejecución de pruebas 42](#_Toc195228265)

[Pruebas Realizadas: 42](#_Toc195228266)

[Prueba de Sensores 42](#_Toc195228267)

[Prueba de App Móvil 42](#_Toc195228268)

[Prueba de Integración Parcial 43](#_Toc195228269)

[XXXV. Evidencia de monitoreo y control del Sprint 43](#_Toc195228270)

[XXXVI. Evidencia de liberación MVP 44](#_Toc195228271)

[Hardware: 44](#_Toc195228272)

[DynamoDB: 45](#_Toc195228273)

[Aplicación movil: 46](#_Toc195228274)

[XXXVII. Problemas Abiertos 47](#_Toc195228275)

[Bibliografía 49](#_Toc195228276)

# Introducción

He tenido varias conversaciones con mi vecino apicultor Roberto, él ha estado gran parte de su vida en el campo y siempre me ha estado contando sobre los problemas de sus abejas, y su gran disminución a lo largo del tiempo, muchas veces hemos discutido sobre la implementación de tecnología en variadas actividades de su rubro y un día me presento con una idea para sus cajones de abejas, propuesta que coincidió con el transcurso de mi carrera universitaria.

Pese a que hay múltiples factores en la disminución de la población de abejas, el mayor factor atribuible a la perdida de colmenas son los constantes cambios de temperatura y humedad dentro de los cajones de las colmenas.

Este proyecto consiste en la creación de dispositivos capaces de medir humedad y temperatura de colmenas de abejas para actividades de apicultura enfocado a grupos de baja a mediana envergadura con la posibilidad de escalado, la razón de existencia de este proyecto es poder ofrecer equipamiento útil para la recopilación de información que potencialmente puede ayudar a cientos de colmenas de abejas y, por ende, el negocio de muchos apicultores a un precio asequible.

# Fundamentación del Problema

Cada día vemos que se escuchan noticias de la próxima disminución significativa de la población de abejas. Esto debido a múltiples factores, entre estos, la contaminación ambiental, cambio climático, disminución de fuentes de alimentos para las abejas, contaminantes de pesticidas en áreas de cultivo o incluso la inclusión de animales exóticos, pese a que no se sabe con exactitud la cantidad de abejas que murieron por la influencia humana, se sabe que hay una tendencia a la disminución de ellas (Latam, 2021). Las condiciones climáticas impredecibles, como las olas de calor o las heladas tempranas, afectan tanto el comportamiento de las abejas como la disponibilidad de flores y recursos alimentarios. Las abejas, esenciales para la polinización de cultivos, enfrentan riesgos cada vez mayores debido a la destrucción de su hábitat natural y la contaminación, lo que hace aún más necesario un sistema de monitoreo para mejorar las condiciones de las colmenas.

El problema que abarcare es sobre la cantidad de humedad y sobre la temperatura de los cajones de abejas, los cuales han presentado problemas de baja población debido a las heladas y gran cantidad de humedad en el ambiente de este último año. Según estudios del Instituto Nacional de Tecnología Agropecuaria (INTA), factores como las condiciones climáticas extremas, la humedad elevada y las enfermedades parasitarias tienen un impacto negativo significativo en las colonias de abejas debilitando a las abejas adultas y afectando la productividad de la colonia. Monitorear estos factores es esencial para la salud de las colmenas y la producción apícola.

La nosemosis es una de las enfermedades que están asociadas a la cantidad de humedad en la colonia, la causante es un parasito llamado nosema spp que se parasita en el intestino de las abejas melíferas adultas y es mortal en su forma aguda lo que causa una gran pérdida económica para el apicultor, para evitar esta enfermedad se necesita controlar la cantidad de humedad del cajón en la cual disminuye el medio por el cual el Imagen que contiene alimentos, tela, cortina

Descripción generada automáticamenteparasito se procrea (Higes, 2010).

Ilustración 1: Signos de diarrea dentro de colmena causada por nosemosis

Imagen que contiene panal, alimentos, alfombra

Descripción generada automáticamenteLa temperatura de igual manera afecta en la salud de las abejas, idealmente una colmena debe cerca de los 35 grados para que se mantenga una estable cantidad de crías (Today, 2021) en caso de que sea más baja puede afectar negativamente la estabilidad de la colmena al tener que enfocar sus esfuerzos en mantener la temperatura dentro del margen ideal descuidando otras funciones de la colmena, por ejemplo, el cuidado de las crías, guardia de la entrada de la colmena o la producción de miel (Langowska, y otros, 2016).

Ilustración 4: Colmena acurrucándose en el centro del cajón para mantener calor

Ilustración 3: Abejas aleteando sus alas en la entrada para enfriar la colmena

Ilustración 2: Diarrea en la entrada del cajón, aumentando la infección a otras abejas

Roberto me cuenta que en casos donde la temperatura es menor, solo recubre todos sus cajones para aumentar el calor sin saber si es lo suficiente o si es necesario, pero le es imposible saber cuánta humedad hay en la colmena hasta que incline el cajón para sacar el agua estancada, por esta razón un sensor de humedad y temperatura es ideal para poder tomar acciones que permitan normalizar la situación en la colmena y faciliten el bienestar de esta.

El monitoreo ambiental fuera de las colmenas también es una práctica importante que, según Apidologie, permite anticipar cuándo será necesario intervenir dentro de la colonia. Al medir la temperatura y humedad del entorno, los apicultores pueden estimar si las colmenas necesitarán aislamiento adicional en el invierno o si se debe tomar acción frente a la aparición de enfermedades relacionadas con el clima. Este enfoque preventivo no solo mejora la salud de las abejas, sino que también optimiza la producción de miel.

# Contexto

Este proyecto está orientado a apicultores de pequeña y mediana escala, quienes buscan optimizar el estado de salud de sus colonias de abejas y aumentar sus niveles de producción. Actualmente, estos apicultores enfrentan diversas dificultades para gestionar y monitorear de manera eficiente sus colmenas, lo que impacta directamente en la productividad y sostenibilidad de su actividad.

# Situación actual

Como se había mencionado antes, los sistemas de monitoreo de colmenas actuales son de precio muy elevado para el público de pequeña y mediana escala, llegando al rango de €300 a €650 por cada cajón, en el caso del mercado nacional podemos encontrar un proyecto similar con fecha de 2019, sin embargo, el proyecto fue desde entonces cerrado.

# Técnica identificación de problemática

Para la identificación de los problemas asociados a la actividad apícola, se realizaron entrevistas a apicultores de la zona con el objetivo de conocer sus principales problemáticas y necesidades en el manejo de sus colmenas. Además, se recopiló información relevante sobre enfermedades de las abejas y otros procesos productivos, gracias a la colaboración de Roberto con alta experiencia en el rubro, complementando esta investigación con información obtenida a través de medios especializados y recursos disponibles en la web.

# Diagrama El contenido generado por IA puede ser incorrecto.Descripción de causas de la problemática

Ilustración 5: Diagrama de Ishikawa

### Hombre

* Existe un bajo nivel de conocimiento en tecnologías emergentes por parte de los apicultores, lo que limita la adopción de herramientas que podrían optimizar sus procesos. Además, deben realizar un trabajo excesivo al no contar con información precisa sobre el estado de sus colmenas, lo que implica inspecciones manuales frecuentes y demandantes.

### Máquina

* El rubro apícola presenta una baja oferta de herramientas tecnológicas orientadas a facilitar labores de monitoreo y gestión, generando una escasa incorporación de soluciones automatizadas en el proceso productivo.

### Entorno

* El mercado actual ofrece pocas alternativas tecnológicas específicas para apicultores de pequeña y mediana envergadura, lo que dificulta encontrar soluciones accesibles y adaptadas a sus necesidades.

### Material

* Los apicultores generalmente disponen de recursos económicos limitados, lo que restringe la posibilidad de adquirir implementos o tecnologías que permitan mejorar la gestión de sus colmenas.

### Método

* Los métodos tradicionales de prevención y control sanitario de las colmenas se realizan sin contar con datos o información previa, lo que disminuye su efectividad y aumenta el riesgo de errores o pérdidas.

### Medio Ambiente

* Los constantes cambios de temperatura y humedad propios del entorno afectan directamente a las abejas y a las colmenas, generando condiciones de estrés que repercuten en la salud de las colonias y en la producción de miel.

# Alcance del proyecto

Este proyecto contempla el desarrollo de una solución tecnológica orientada a apoyar la gestión y monitoreo de colmenas para apicultores de pequeña y mediana escala. El sistema permitirá registrar información relevante sobre las colmenas, monitorear parámetros ambientales, y entregar alertas preventivas sobre posibles problemas o anomalías.

## El proyecto considera:

* Desarrollo de una aplicación web y/o móvil para la gestión de colmenas.
* Registro de información básica de las colmenas (mediciones y ubicación).
* Visualización de datos sobre temperatura y humedad del entorno e interior.
* Generación de alertas frente a condiciones desfavorables.
* Diseño de una interfaz amigable para usuarios con bajo conocimiento tecnológico.
* Entrega de reportes básicos de estado de las colmenas.

## Fuera del alcance:

* Gestión de inventarios de productos apícolas (miel, cera, etc.).
* Procesos de venta o comercialización de productos apícolas.
* Monitoreo en tiempo real de enfermedades específicas de las abejas.
* Sensores fuera del alcance de placas de desarrollo de bajo consumo.

# Objetivo general

Desarrollar un sistema de monitoreo remoto para colmenas apícolas, que permita registrar y transmitir información ambiental relevante, como temperatura, humedad interna y externa de los cajones de abejas, junto con la ubicación geográfica de las colmenas, mediante un dispositivo de bajo consumo energético y conectividad 4G, visualizando la información a través de una aplicación móvil orientada a apicultores de pequeña y mediana escala.

# Objetivos específicos

* Diseñar un dispositivo de monitoreo basado en una placa de desarrollo de bajo consumo energético, capaz de obtener datos de temperatura y humedad interna y externa de las colmenas.
* Implementar un sistema de transmisión de datos a través de conectividad móvil, permitiendo enviar la información recopilada desde las colmenas hacia un servidor o plataforma en la nube.
* Desarrollar una aplicación móvil que permita a los apicultores visualizar en tiempo real los datos de sus colmenas, incluyendo parámetros ambientales y ubicación geográfica.
* Integrar un sistema de geolocalización mediante GPS para registrar y mostrar la ubicación exacta de las colmenas monitoreadas.
* Validar y probar el funcionamiento del sistema desarrollado en un entorno controlado, evaluando la precisión de los datos obtenidos y la usabilidad de la aplicación móvil.

# Métricas de objetivos

|  |  |  |
| --- | --- | --- |
| Objetivo Específico | Métrica Asociada | Criterio de Éxito |
| Diseñar dispositivo de monitoreo | Dispositivo funcional capaz de medir temperatura y humedad | Medición estable y confiable de parámetros ambientales durante al menos 48 horas continuas |
| Implementar sistema de transmisión de datos 4G | Envío de datos desde dispositivo a servidor | Datos transmitidos exitosamente con una frecuencia mínima de actualización cada 5 minutos |
| Desarrollar aplicación móvil | Visualización de datos en app | App funcional, capaz de mostrar información de al menos 1 colmena monitoreada en tiempo real |
| Integrar geolocalización GPS | Registro y visualización de ubicación de colmenas | Precisión de ubicación menor a 10 metros de desviación respecto a ubicación real |
| Validar funcionamiento del sistema | Pruebas de funcionamiento en entorno controlado | Sistema operando sin fallos críticos en un periodo mínimo de prueba de 1 semana |

# Alternativas de solución

## Mediciones manuales

Crear dos dispositivos, uno que mida la humedad y temperatura del ambiente y otro portátil capaz de realizar mediciones al ser insertado en las aberturas de los cajones

### Ventajas

* Bajo costo
* No conocimientos tecnológicos necesarios

### Desventajas

* Alto consumo de tiempo y esfuerzo
* La información no es en tiempo real
* Mayor pérdida de colmenas por detección tardía

## Sistema cableado

Instalación de sensores en las colmenas, conectados por cables a un sistema central de registro ubicado cerca del apiario

### Ventajas

* Mediciones más precisas y constantes
* Sin dependencia de cobertura de redes móviles

### Desventajas

* Instalación compleja
* Depende mucho de la geografía del sector
* Alto costo

## Sistema de monitoreo remoto (solución propuesta)

Creación de dispositivo inalámbrico, autónomo y de bajo consumo, capaz de realizar mediciones de humedad y temperatura dentro y fuera de los cajones

### Ventajas

* Monitoreo en tiempo real desde cualquier lugar
* Sin cableado
* Escalable

### Desventajas

* Costo asociado al hardware y conectividad
* Dependencia de señal móvil

# Alcances y limitaciones del proyecto

Este proyecto está orientado a desarrollar un sistema de monitoreo remoto de colmenas de abejas mediante el uso de placas de desarrollo de bajo consumo que transmiten humedad, temperatura y ubicación geográfica de las colmenas mediante redes móviles a una aplicación móvil

El desarrollo de este proyecto considera:

* Desarrollo del hardware, siendo este una placa de desarrollo, módulo de redes móviles, sensores de humedad y temperatura y su propio enclosure para protección del ambiente.
* Desarrollo de software, para la visualización de los datos en tiempo real de forma intuitiva para el usuario.
* Pruebas de funcionamiento, para verificación del funcionamiento del dispositivo.
* Servicio de almacenamiento en la nube, para almacenamiento de datos.

No incluye:

* Sensores que superen los límites de la placa de desarrollo como pesas, cámaras o sistemas con inteligencia artificial.

El proyecto se ve limitado por:

* La cobertura de las redes móviles en zonas rurales.
* Costo inicial del proyecto, pese a que es de menor costo que otras opciones, el costo puede ser algo elevado para apicultores con recursos limitados, sobre todo por la implementación de redes móviles.
* Vida útil y mantenimiento, el dispositivo debe ser revisado una vez por mes para asegurarse que sigue funcionando correctamente y para reemplazo de baterías.
* Dependencia de un dispositivo móvil, limitante para personas sin smartphones, aunque sea una estadística en descendencia, también existe el problema de la poca familiaridad con dispositivos inteligentes

# Factibilidades

## Factibilidad técnica

El proyecto dispone de tecnologías accesibles como lo son las placas de desarrollo (por ejemplo, el ESP32 o ESP8266) utilizadas en proyectos de IoT, estos permiten la creación de dispositivos de bajo consumo energético que transmiten información de manera eficiente, también existen sensores altamente fiables para la toma de datos compatibles con estas placas de desarrollo que se integran de manera fácil y rápida.

El uso de redes permite la transmisión de datos de forma inalámbrica y su área de cobertura es cada vez más ampliada alrededor del país.

También existen herramientas y frameworks para el desarrollo de aplicaciones móviles que permiten crear interfaces de usuario intuitivas y funcionales.

## Factibilidad económica

Desde el punto de vista económico el proyecto tiene un costo relativamente bajo considerando las otras opciones similares, los sensores y placas de desarrollo son de bajo costo y el módulo de red móvil evita el malgasto en infraestructura costosa, puede que sea una inversión considerable pero los beneficios que aporta como la reducción de tiempos de inspección, prevención de pérdidas y aumento en la productividad generara un retorno de inversión positivo a los apicultores a mediano y largo plazo.

También hay que considerar que el proyecto es fácilmente escalable y se puede expandir a otros grupos de colmenas ya que cada dispositivo transmite información de forma independiente desde una sola aplicación, solo bastaría modificar una cierta parte del código.

## Factibilidad Operativa

La aplicación móvil estaría diseñada para ser sencilla de ocupar, permitiendo a apicultores visualizar y gestionar colmenas sin necesidades de conocimientos técnicos

El sistema no interfiere con las rutinas de los apicultores al ser independiente y pequeño en comparación a otros dispositivos

El mantenimiento del dispositivo será mínimo y fácil de ejecutar, la batería es fácilmente reemplazable y recargable, y gran parte del hardware es modular y por tanto fácil de reemplazar

# Propuesta de solución

La propuesta de solución consiste en el desarrollo de un sistema de monitoreo remoto para colmenas apícolas, que permitirá a los apicultores gestionar y supervisar sus colmenas de forma eficiente y sin necesidad de intervención física constante. Este sistema estará compuesto por un dispositivo autónomo de bajo consumo energético, que medirá los parámetros ambientales (temperatura y humedad) dentro y fuera de las colmenas, y transmitirá estos datos en tiempo real a través de una red móvil a una aplicación móvil.

Componentes del Sistema:

* Dispositivo de Monitoreo:  
  El dispositivo será un sensor autónomo, integrado con un microcontrolador de bajo consumo energético (ESP32 o ESP8266), un módulo de red SIM7600SA o similar, sensor de humedad y temperatura SHT41(interior por su tamaño) y DHT22 para medir el ambiente dentro y fuera de la colmena. Este dispositivo operará con batería y podrá funcionar de manera continua durante largos períodos sin necesidad de recarga frecuente (aproximadamente un mes).
* Aplicación Móvil:  
  La aplicación móvil permitirá a los apicultores visualizar en tiempo real los datos sobre sus colmenas, tales como la temperatura, la humedad, y la ubicación geográfica de cada colmena. La app también enviará alertas a los usuarios cuando los parámetros ambientales se desvíen de los valores óptimos para el bienestar de las abejas, permitiendo la toma de decisiones informadas. Además, contara con funciones cloud para almacenamiento de datos de los sensores
* Conectividad móvil:  
  La comunicación de datos será realizada a través de una red móvil, permitiendo que el sistema funcione incluso en áreas rurales o apartadas, sin depender de redes Wi-Fi locales. Esta tecnología garantizará que los datos sean enviados de manera confiable y en tiempo real.
* Interfaz de Usuario:  
  La app estará diseñada para ser intuitiva y de fácil uso, permitiendo que apicultores con poca experiencia tecnológica puedan gestionar el monitoreo de sus colmenas de manera eficiente. Además, la aplicación permitirá visualizar gráficas y reportes históricos para un mejor seguimiento del estado de las colmenas.

Ventajas de la Solución:

* Monitoreo remoto y en tiempo real: Los apicultores podrán consultar los datos en cualquier momento y desde cualquier lugar, lo que les permitirá realizar ajustes de manera rápida y evitar problemas graves.
* Reducción de costos operativos: Al reducir la necesidad de inspecciones físicas constantes, se ahorrará tiempo y esfuerzo, lo que aumentará la productividad y reducirá la posibilidad de pérdidas debido a condiciones desfavorables.
* Facilidad de uso: La solución está diseñada para ser accesible, sin requerir conocimientos técnicos avanzados, lo que facilita su adopción por apicultores con diversos niveles de experiencia.
* Escalabilidad: El sistema es flexible y se puede ampliar fácilmente para gestionar más colmenas o apicultores en el futuro, sin necesidad de realizar cambios significativos en la infraestructura.

Justificación de la Solución:

Esta solución fue elegida debido a su adaptabilidad y viabilidad para apicultores de pequeña y mediana escala, que enfrentan dificultades con el monitoreo manual de sus colmenas. A través de la automatización del proceso y la transmisión remota de datos, se mejora la eficiencia operativa y la sostenibilidad de las prácticas apícolas. Además, el uso de conectividad inalámbrica garantiza que incluso los apicultores ubicados en zonas remotas puedan aprovechar la tecnología sin depender de infraestructura de red compleja.

# Diagrama de contexto

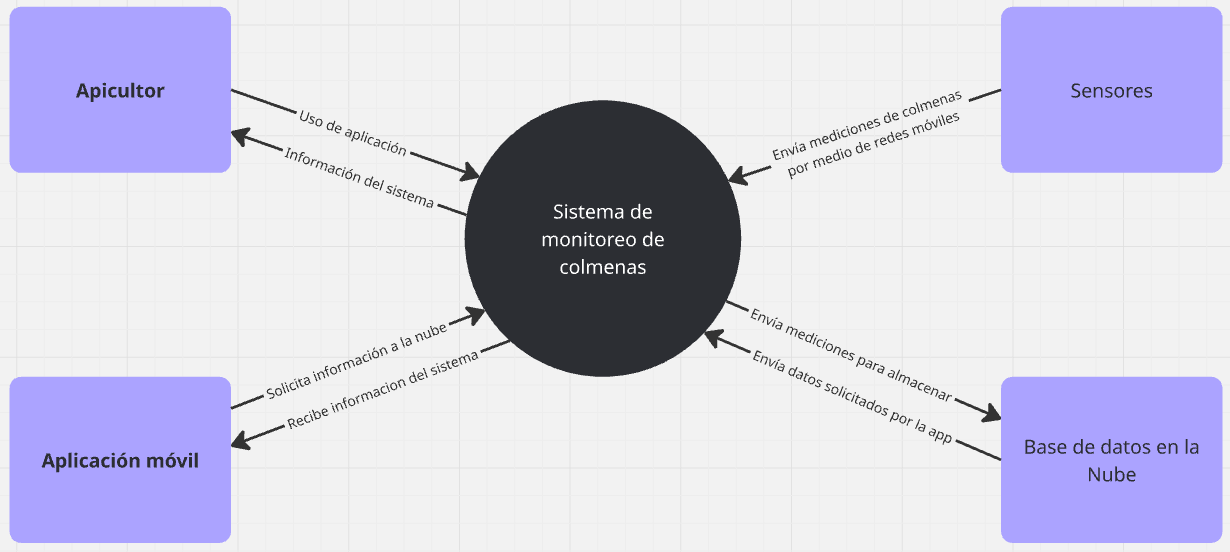


Ilustración 6: Diagrama de contexto

# Plan de proyecto

|  |  |  |
| --- | --- | --- |
| 1. Fase | Actividad Principal | Fecha estimada |
| Investigación | Levantamiento de información, problemáticas y referentes. | 19 marzo - 04 abril |
| Integración Backend | Vincular app con base de datos y mostrar datos en app. | 04 - 19 abril |
| Ajustes Hardware | Optimización de dispositivos y posible migración a datos móviles con chip. | 20 - 28 abril |
| Pruebas Generales | Probar flujo completo de datos entre hardware y app. | 29 abril - 10 mayo |
| Optimización | Mejoras en diseño de app, interfaz y rendimiento. | 11 - 19 mayo |
| Documentación | Redacción de informe final y preparación presentación. | 20 - 31 mayo |
| Pruebas Finales | Test en entorno real (campo o simulado) y cierre proyecto. | 01 - 06 junio |

# Metodología de desarrollo

Debido a que es un proyecto con software y hardware, un método de gestión por Scrum es ideal, este método a que presenta beneficios para los desarrolladores y usuarios permitiendo comunicación entre ellos, facilitando la entrega de avances de manera iterativa e incremental, adaptación de cambios y necesidades del usuario, simplificación de la gestión de tiempos y tareas, y el trabajo está mejor organizado por la utilización de sprints y backlogs.

# Metodología de desarrollo

La herramienta de gestión de Scrum que se ocupara es Jira, la cual es una plataforma web muy útil, simple y comprensible que me permite crear sprints en base a historias de usuario, vincular repositorios de mi código con los avances en el sprint y demostrar avances del proyecto de forma clara.

|  |  |
| --- | --- |
| Elemento | Aplicación en el Proyecto |
| Sprint | Períodos de trabajo de 4 semanas. |
| Backlog | Lista de funcionalidades e historias de usuario. |
| Sprint Backlog | Selección de tareas específicas que se llevaran a cabo cada Sprint. |
| Reuniones de control | Dos veces por semana se harán revisión de avances y planificación de próximas tareas. |
| Roles | Desarrollador (Yo), Apicultor (Usuario). |

# Plan de monitoreo y control

Para monitorear los avances del proyecto se realizarán dos reuniones semanales donde se mostrarán los avances del proyecto, durante dichas reuniones se repasará el seguimiento del cumplimiento de tareas designadas en Jira, se validará las funcionalidades implementadas durante ese lapso y se realizará un análisis de posibles desviaciones con respecto a la planificación del proyecto, todo siendo documentado e inscrito en este documento.

# Plan de gestión de riesgos

|  |  |  |  |
| --- | --- | --- | --- |
| Riesgo identificado | Probabilidad | Impacto | Plan de mitigación |
| Fallas de comunicación de datos móviles en zonas rurales | Media | Alto | Reintento de conexión automático |
| Fallo de sensores o de hardware | Media | Alto | Contar con sensores de repuesto y realizar pruebas de funcionamiento previo a instalación |
| Perdida de datos en la base de datos | Baja | Alto | Respaldos periódicos en la memoria del dispositivo donde vinculada la app |
| Retraso en el desarrollo de la app | Media | Medio | Priorización de tareas críticas y planificación flexible en sprints |
| Daños físicos en los dispositivos por ambiente, animales o robos | Baja | Alto | Realizar enclosures protectoras, a prueba de agua y visualmente compactas para su anclaje en cajones |
| Consumo energético mayor al esperado en los dispositivos | Media | Medio | Realizar pruebas de uso de baterías previo a test en campo y optimización de código para reducción de consumo |
| Falta de disponibilidad de componentes | Baja | Medio | Adquirir anticipadamente piezas necesarias para crear un stock de respaldo |
| Dificultad de adopción de la app por parte de los usuarios | Media | Medio | Tener en cuenta un diseño simple e intuitivo durante el desarrollo de la aplicación y contar con feedback de los early adopters |
| Errores de precisión en las mediciones de los sensores | Media | Medio | Calibrar sensores previos a instalación en un ambiente controlado y validado con instrumentos separados |

# Diseño de proyecto de alto nivel

Duración total

El proyecto tiene una duración total desde el 19 de marzo hasta el 6 de junio, con tareas específicas distribuidas a lo largo de estas semanas.

## Fases y tiempos de entrega:

* Investigación de información y compra de materiales (19 marzo – 4 abril)
* Desarrollo, integración del backend con la base de datos y armado de prototipos de prueba (08 - 11 abril) – Hito 1
* Optimización y construcción de hardware y software (11 - 28 abril)
* Pruebas generales del funcionamiento de la aplicación (29 abril – 9 mayo) – Hito 2
* Optimización de la app y mejoras de rendimiento (10 - 19 mayo)
* Redacción del informe final y preparación de la presentación (20 - 31 mayo)
* Pruebas finales en entorno real y cierre del proyecto (01 - 06 junio) – Hito 3

## Recursos necesarios:

* Humanos: Desarrollador (yo), Supervisor (profesores), Apoyo de pruebas de campo (Vecino apicultor).
* Materiales: Dispositivos de hardware (ESP32/ESP8266, sensores, etc.), recursos para pruebas de campo, equipos de desarrollo para la app móvil, acceso a plataformas de prueba (Mi dispositivo Android y emuladores).
* Tecnológicos: Plataformas de desarrollo (Flutter) y base de datos (AWS DynamoDB, AWS Lambda, AWS, IoT Core).
* Financieros: Presupuesto estimado para los componentes del hardware.

## Hitos:

* Hito 1: Investigación de requerimientos, planeación de proyecto e inicio del desarrollo.
* Hito 2: Finalización de la app móvil lista para conectar con los dispositivos.
* Hito 3: Completar las pruebas generales, pruebas finales, entrega final y presentación.

## Dependencias:

* El desarrollo del backend depende de la integración de hardware.
* Las pruebas de la app móvil dependen de la correcta conexión con la base de datos y de los dispositivos funcionando correctamente.
* La realización de pruebas en campo depende de la disponibilidad de mi vecino quien ofrece ayuda según su disponibilidad

## Recursos adicionales:

* Capacitación: Si es necesario, crear tutoriales o entrenar a los apicultores sobre cómo utilizar la app de forma efectiva.
* Pruebas de campo: Asegurarse de que los dispositivos funcionen correctamente en condiciones reales y recojan datos precisos.

# Vista lógica

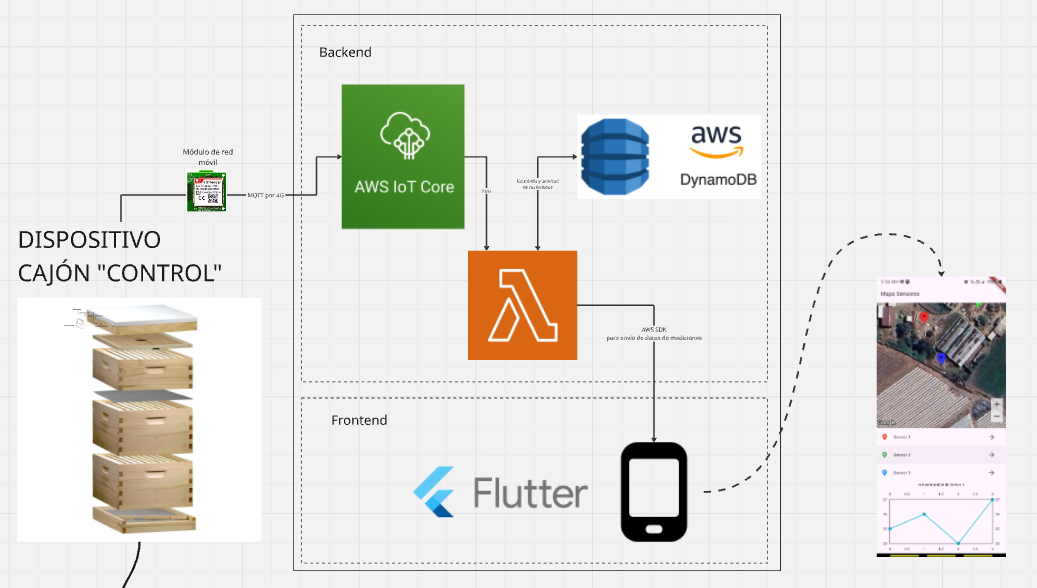


Ilustración 7: Vista lógica

# Vista física

Dispositivo de prueba actual (D1 mini sin módulo de redes móviles) Un circuito electrónico

El contenido generado por IA puede ser incorrecto.

Ilustración 8: Vista física del dispositivo de prueba

## Dispositivo final esperado (ESP32 con módulo de red)

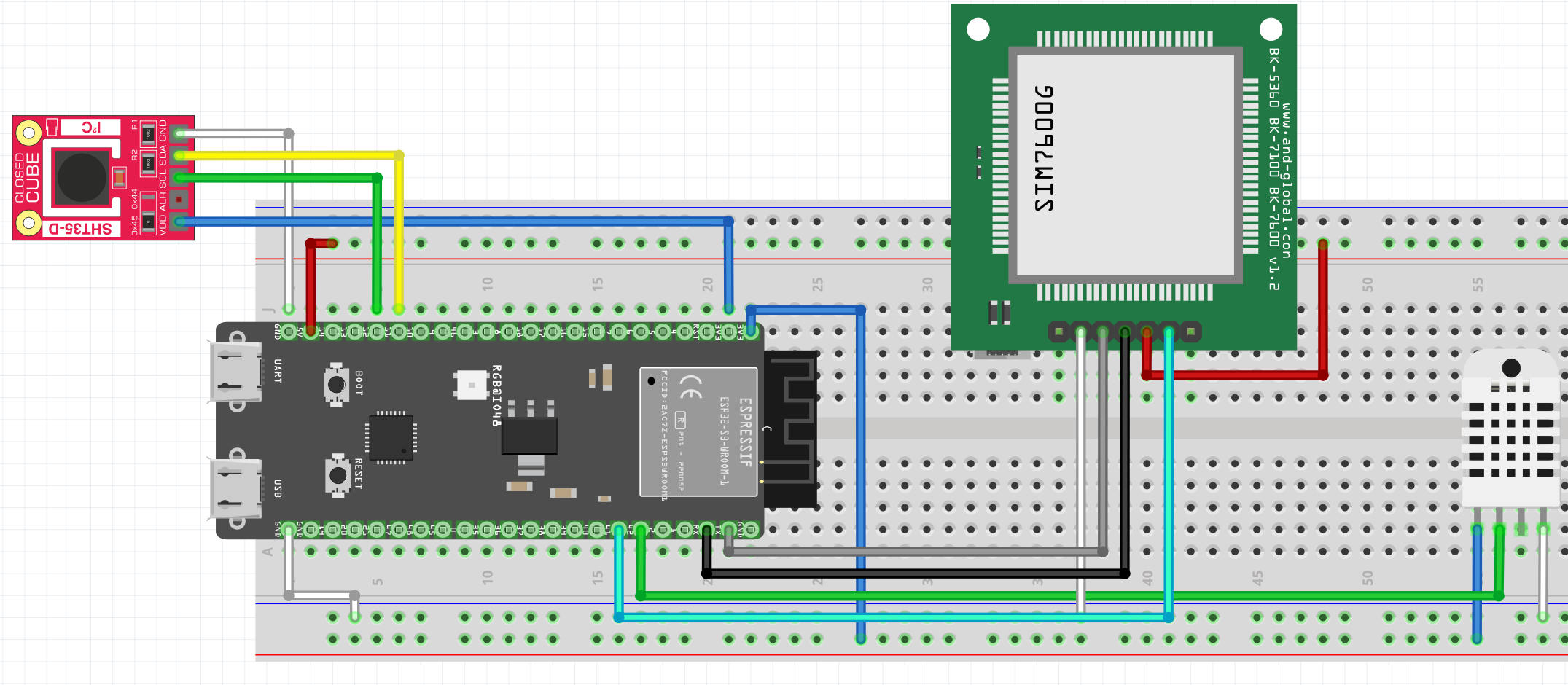


Ilustración 9: Vista física del dispositivo final

# Vista de despliegue

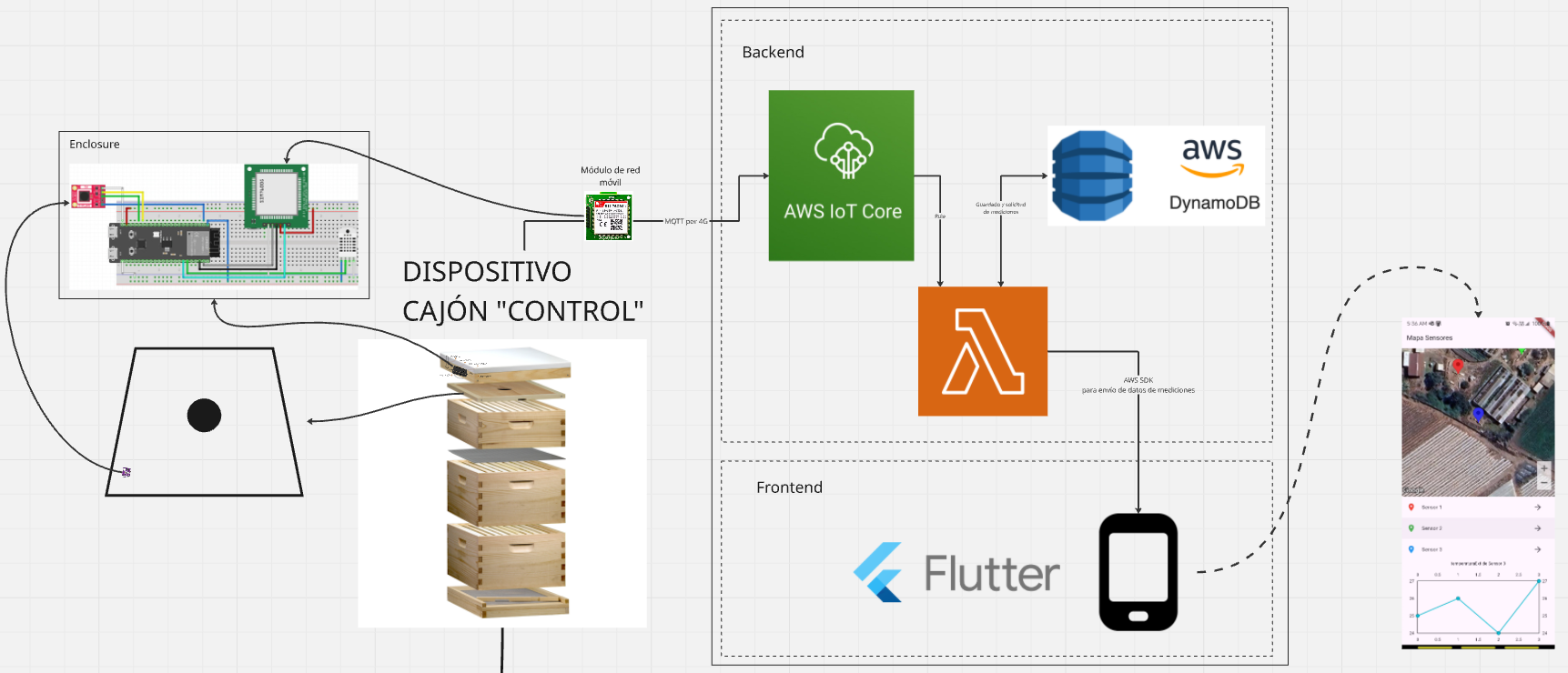


Ilustración 10: Vista de despliegue

# Vista de escenarios

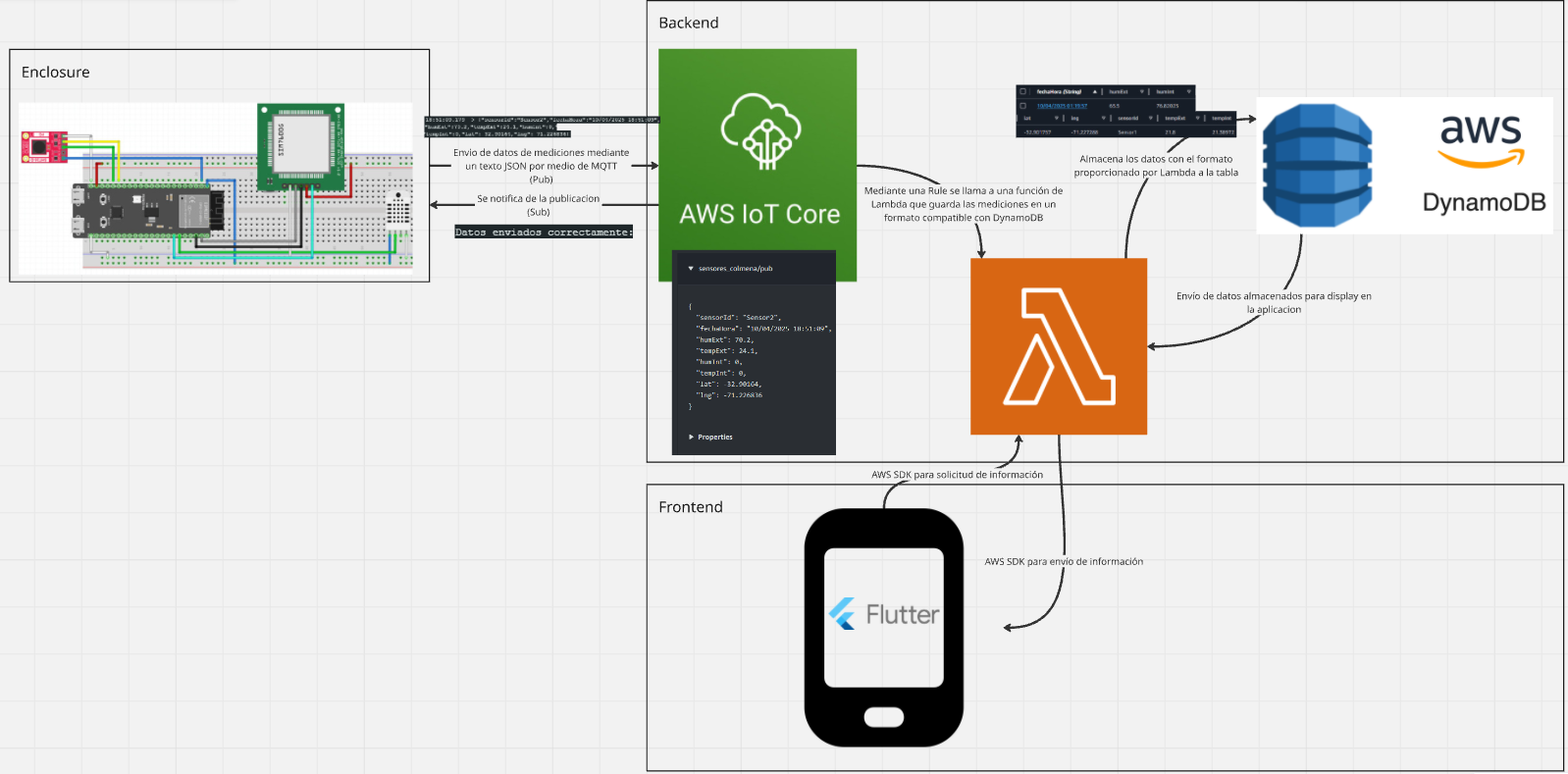


Ilustración 11: Vista de escenarios

# Iteración 1

El objetivo principal de esta iteración es crear una versión funcional básica del sistema que permita la recopilación de datos de los sensores (temperatura y humedad), su envío por WiFi, la creación de la aplicación para implementar la base de datos y la visualización de estos datos en el database. Se pretende verificar el funcionamiento del sistema y hacer ajustes iniciales.

#### **Tareas Específicas:**

**Armado del dispositivo:**

* + Conectar el sensor **DHT22 y SHT41** al **D1 Mini**.
  + Configurar la lectura de los datos de temperatura y humedad.
  + Realizar pruebas de lectura de los sensores y depuración de la señal.

**Desarrollo del firmware del ESP32/D1 Mini:**

* + Escribir el código para leer los datos de los sensores.
  + Configurar la conexión WiFi en el ESP32/D1 Mini.
  + Transmitir los datos a través de la red WiFi por MQTT a AWS IoT Core.

**Desarrollo de la aplicación móvil:**

* + Crear la interfaz básica en la aplicación móvil para visualizar los datos de temperatura, humedad y geolocalización en un mapa.
  + Mostrar los datos en gráficos simples.

**Pruebas de conectividad entre servicios:**

* + Verificar que el **ESP32/D1 Mini** se conecta correctamente a los servicios de AWS y que estos guarden satisfactoriamente los datos a DynamoDB.

**Pruebas de funcionamiento básico:**

* + Hacer pruebas de lectura y transmisión de datos en un ambiente controlado.
  + Verificar la estabilidad de la conexión y la actualización en tiempo real de la aplicación móvil.

# Duración y meta del Sprint

La iteración 1 tiene una duración de 3 semanas desde el inicio de operaciones. Durante este tiempo, se realizó la investigación, implementación y prueba de las funcionalidades básicas. Los hitos clave incluyen:

* Semana 1: Investigación y entrevistas a apicultores de la zona (19– 26 marzo).
* Semana 2: Compra de piezas necesarias para dispositivo de pruebas e investigación del desarrollo de la aplicación (26 de marzo – 4 de abril).
* Semana 3: Desarrollo de aplicación de prueba, firmware, database e inicio de documentación (7 – 10 de abril).

La meta de este sprint es presentar el sistema y dejar una base para las siguientes iteraciones.

# Funcionalidades para realizar

|  |  |  |  |
| --- | --- | --- | --- |
| Nº | Funcionalidad | Descripción | Estado Esperado |
| 1 | Lectura de Sensores | Lectura de temperatura y humedad desde los sensores DHT22 y SHT41 conectados al D1 Mini | Los datos deben obtenerse correctamente cada 10 minutos |
| 2 | Envío de Datos por WiFi | Enviar los datos leídos a los servicios de AWS por medio de la conexión WiFi | Comunicación estable y envío exitoso de los datos a la nube o base de datos local |
| 3 | Visualización de DynamoDB | Mostrar los datos recibidos (temperatura y humedad, interna y externa y latitud longitud) en tiempo real en la base de datos | Datos actualizados automáticamente en DynamoDB |
| 4 | Gestión Básica de Conexión | Gestión de la reconexión automática del dispositivo en caso de pérdida de conexión WiFi | El sistema debe reconectarse sin intervención manual |

# Historias de usuario a realizar

Las historias de usuario registrados en Jira son las siguientes:

* Yo como apicultor, quiero chequear la temperatura y humedad dentro de mis colmenas.
* Yo como apicultor, quiero chequear la temperatura y humedad alrededor de mis colmenas.
* Yo como desarrollador, quiero registrar los datos de los sensores de las colmenas a un database en la nube.
* Yo como apicultor, quiero recibir alertas cuando la humedad o la temperatura este sobre o bajo los límites.
* Yo como apicultor, quiero ver los datos históricos de la temperatura y humedad
* Yo como apicultor, quiero que el dispositivo trabaje de forma independiente de una red Wifi.
* Yo como apicultor, quiero una aplicación simple o un dashboard.
* Yo como desarrollador, quiero crear una aplicación que muestre los datos del sensor en un formato fácil de leer.
* Yo como desarrollador, quiero asegurarme de que los dispositivos sean de bajo consumo.
* Yo como apicultor, quiero encontrar la ubicación de los cajones controlados en un mapa.
* Yo como apicultor, quiero que sea simple cambiar la batería de los dispositivos.
* Yo como desarrollador, necesito crear una enclosure, aprueba de agua que permita proteger todos los componentes de los elementos.
* Yo como apicultor, necesito ver el estado del dispositivo de forma física para verificar que no tenga problemas.
* Yo como apicultor, quiero una aplicación simple que me muestre los datos absolutamente necesarios para el cuidado de colmenas.
* Yo como apicultor, quiero tener la posibilidad de ver gráficos asociados a las mediciones de las colmenas.

# Definición Sprint backlog

Sprint 1 (19 marzo – 11 abril):

* Yo como apicultor, quiero tener la posibilidad de ver gráficos asociados a las mediciones de las colmenas.
* Yo como apicultor, quiero chequear la temperatura y humedad dentro de mis colmenas.
* Yo como apicultor, quiero chequear la temperatura y humedad alrededor de mis colmenas.
* Yo como desarrollador, quiero registrar los datos de los sensores de las colmenas a un database en la nube.
* Yo como apicultor, quiero encontrar la ubicación de los cajones controlados en un mapa.

Sprint 2 (11 abril – 9 mayo):

* Yo como apicultor, quiero recibir alertas cuando la humedad o la temperatura este sobre o bajo los límites.
* Yo como apicultor, quiero ver los datos históricos de la temperatura y humedad.
* Yo como apicultor, quiero una aplicación simple o un dashboard.
* Yo como desarrollador, quiero crear una aplicación que muestre los datos del sensor en un formato fácil de leer.
* Yo como apicultor, quiero que el dispositivo trabaje de forma independiente de una red Wifi.

Sprint 3 (9 mayo – 6 junio):

* Yo como desarrollador, quiero asegurarme de que los dispositivos sean de bajo consumo.
* Yo como apicultor, quiero que sea simple cambiar la batería de los dispositivos.
* Yo como desarrollador, necesito crear una enclosure, aprueba de agua que permita proteger todos los componentes de los elementos.
* Yo como apicultor, necesito ver el estado del dispositivo de forma física para verificar que no tenga problemas.

# Diseño detallado

El proyecto consiste en el dispositivo, la aplicación y los servicios AWS.  
Creación del dispositivo de prueba:

Un circuito electrónico

El contenido generado por IA puede ser incorrecto.

Ilustración 13: Vista física del dispositivo de prueba

* Wemos D1 Mini: Placa de desarrollo que utiliza una conexión a wifi para transmitir información, eficientes y baratos, los cuales enviaran la información de los sensores a los servicios de AWS.

Interfaz de usuario gráfica

Descripción generada automáticamenteEl precio de cada unidad es de $2.990.

Ilustración 14: Wemos D1 Mini

* DHT22: Sensor de humedad y temperatura para medir fuera de la colmena.

Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza bajaPrecio de cada unidad es de $1.990.

Ilustración 15: DHT22

* SHT41: Sensor de humedad y temperatura más pequeño que DHT22 para medir al interior de la colmena.

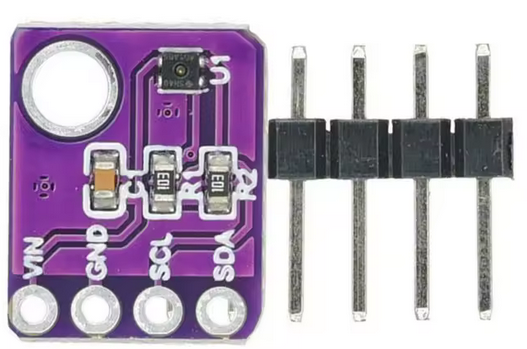
Precio de cada unidad es de $2.500

Ilustración16: SHT41

|  |  |
| --- | --- |
| Costo total del dispositivo: | $7.480 |
| Total de dispositivos: | 2 |
| Total de costos: | $14960 |

## Programación del dispositivo de prueba

#include <ESP8266WiFi.h>

#include <WiFiClientSecure.h>

#include <PubSubClient.h>

#include <ArduinoJson.h>

#include <time.h>

#include <Wire.h>

#include "Adafruit\_SHT4x.h"

#include "secrets.h"

#include "DHT.h"

Importación de librerías necesarias para el funcionamiento del dispositivo, incluyendo una librería creada llamada “secrets.h”, esta librería contiene el SSID y contraseña de mi WiFi, certificados, dominio y key de los servicios de AWS

#define TIME\_ZONE -4

#define DHTPIN D5

#define DHTTYPE DHT22

Definición de la zona horaria para una función de tiempo, el pin donde esta conectado el sensor DHT22 y la definición de que el tipo de sensor DHT es un DHT22

DHT dht(DHTPIN, DHTTYPE);

Adafruit\_SHT4x sht4 = Adafruit\_SHT4x();

Inicialización de los sensores, el sensor DHT se tiene que definir el pin donde este conectado y el tipo de DHT, en cambio el SHT siempre tiene que ir conectado en los puertos SCL y SDA, por lo que solo se crea una instancia de SHT4

float h ;

float t;

String sensor\_id = "Sensor2" ;

sensors\_event\_t humidityInt, tempInt;

String fecha;

unsigned long lastMillis = 0;

unsigned long previousMillis = 0;

const long interval = 5000;

const long delayEnvio = 10\*60\* 1000;

Definición de variables a ocupar en el resto del código

h y t representan los valores medidos por el DHT

sensor\_id es el nombre del dispositivo, en este caso, es el segundo dispositivo, ambos dispositivos se conectan al mismo thing en AWS y transmiten al mismo sub de MQTT

humidityInt y tempInt son las variables para el SHT

El resto de las variables son para el envío de mensajes JSON al MQTT de AWS IoT

delayEnvio es el delay para cada medida de los sensores, en este caso realiza medidas cada 10 minutos

#define AWS\_IOT\_PUBLISH\_TOPIC   "sensores\_colmena/pub"

#define AWS\_IOT\_SUBSCRIBE\_TOPIC "sensores\_colmena/sub"

Se define los topic donde se publican los datos y el sub por donde recibe las notificaciones de envío exitoso

WiFiClientSecure net;

BearSSL::X509List cert(cacert);

BearSSL::X509List client\_crt(client\_cert);

BearSSL::PrivateKey key(privkey);

PubSubClient client(net);

Se inicia el cliente del WiFi de la placa, se cargan los certificados y la key de AWS, también se inicia el cliente MQTT

void updateFecha() {

  time\_t now = time(nullptr);

  struct tm timeinfo;

  localtime\_r(&now, &timeinfo);

  char timeStr[20];

  strftime(timeStr, sizeof(timeStr), "%d/%m/%Y %H:%M:%S", &timeinfo);

  fecha = String(timeStr);

}

Función de actualización de fecha, primero obtiene la hora actual, la convierte a el horario local y lo escribe en el formato día/mes/año hora:minuto:segundos y finalmente lo guarda como un string

void NTPConnect(void)

{

  Serial.print("Configurando tiempo por SNTP");

  configTime(TIME\_ZONE \* 3600, 0 \* 3600, "pool.ntp.org", "time.nist.gov");

  now = time(nullptr);

  while (now < nowish)

  {

    delay(500);

    Serial.print(".");

    now = time(nullptr);

  }

  Serial.println("Terminado");

  struct tm timeinfo;

  gmtime\_r(&now, &timeinfo);

  char timeStr[20];

  strftime(timeStr, sizeof(timeStr), "%d/%m/%Y %H:%M:%S", &timeinfo);

  Serial.print("Current time: ");

  Serial.print(asctime(&timeinfo));

  fecha = String(timeStr);

}

La función NTPConnect, primero configura la NTP con la zona horaria para sincronizar la hora del dispositivo

void messageReceived(char \*topic, byte \*payload, unsigned int length)

{

  Serial.print("Recibido [");

  Serial.print(topic);

  Serial.print("]: ");

  for (int i = 0; i < length; i++)

  {

    Serial.print((char)payload[i]);

  }

  Serial.println();

}

Imprime el topico del mensaje y luego imprime el carácter del mensaje recibido

void connectAWS()

{

  delay(3000);

  WiFi.mode(WIFI\_STA);

  WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

  Serial.println(String("Intentando conectar al SSID: ") + String(WIFI\_SSID));

  while (WiFi.status() != WL\_CONNECTED)

  {

    Serial.print(".");

    delay(1000);

  }

  NTPConnect();

  net.setTrustAnchors(&cert);

  net.setClientRSACert(&client\_crt, &key);

  client.setServer(MQTT\_HOST, 8883);

  client.setCallback(messageReceived);

  Serial.println("Conectando a AWS IOT");

  while (!client.connect(THINGNAME))

  {

    Serial.print(".");

    delay(1000);

  }

  if (!client.connected()) {

    Serial.println("AWS IoT Timeout");

    return;

  }

  client.subscribe(AWS\_IOT\_SUBSCRIBE\_TOPIC);

  Serial.println("AWS IoT Conectado");

}

Primero intenta conectar al wifi con los datos de secrets.h y espera a que se conecte, sincroniza la hora NTP, configura los certificados de seguridad para AWS IoT, registra la función callback para mensajes recibidos, luego intenta conectar a AWS IoT con MQTT, si falla lo reintenta cada un segundo y finalmente se subscribe a el tópico definido anteriormente para recibir mensajes

void publishMessage()

{

  updateFecha();

  StaticJsonDocument<300> doc;

  doc["sensorId"] = sensor\_id;

  doc["fechaHora"] = fecha;

  doc["humExt"] = h;

  doc["tempExt"] = t;

  doc["humInt"] = humidityInt.relative\_humidity;

  doc["tempInt"] = tempInt.temperature;

  doc["lat"] = -32.901640;

  doc["lng"] = -71.226836;

  char jsonBuffer[512];

  serializeJson(doc, jsonBuffer);

  if(client.publish(AWS\_IOT\_PUBLISH\_TOPIC, jsonBuffer)){

    Serial.print("Humedad Ext: ");

    Serial.print(h);

    Serial.print("%  Temperatura Ext: ");

    Serial.print(t);

    Serial.println("°C");

    Serial.print("Humedad Int: ");

    Serial.print(humidityInt.relative\_humidity);

    Serial.print("%  Temperatura Int: ");

    Serial.print(tempInt.temperature);

    Serial.println("°C");

    Serial.println("Datos enviados correctamente: ");

    Serial.println(jsonBuffer);

  }else{

    Serial.println("Error de envio");

  }

}

Obtiene la fecha actualizada, crea el documento JSON con las variables previamente definidas, convierte el JSON a un texto y lo publica en AWS

void setup()

{

  Serial.begin(115200);

  connectAWS();

  dht.begin();

  Wire.begin();

  if(!sht4.begin()){

    Serial.println("Error de lectura en el SHT41");

    delay(1);

  }

  Serial.println("Sensor SHT41 encontrado");

  Serial.println("Sensor DHT22 encontrado");

  Serial.println(sensor\_id);

}

En la función Setup inicia comunicación serial por USB (envía mensajes en la consola de Serial Monitor), activa la función de connectAWS, inicia los sensores conectados y posteriormente imprime la id del sensor, en este caso es el sensor 2.

void loop()

{

  h = dht.readHumidity();

  t = dht.readTemperature();

  sht4.getEvent(&humidityInt, &tempInt);

  if (isnan(h) || isnan(t) )

  {

    Serial.println(F("Error de lectura en el DHT22"));

    return;

  }

  now = time(nullptr);

  if (!client.connected())

  {

    connectAWS();

  }

  else

  {

    client.loop();

    if (millis() - lastMillis > delayEnvio)

    {

      lastMillis = millis();

      publishMessage();

    }

  }

}

En la función Loop primero lee el sensor DHT22 y los vincula a sus variables correspondientes, luego hace lo mismo para el sensor SHT41, si hay un error en las lecturas de DHT22 no hace nada en el ciclo, obtiene la hora actual, verifica si se desconecto AWS, si ese es el caso se vuelve a conectar, caso contrario mantiene la conexión MQTT activa, si pasaron 10 minutos actualiza el tiempo del último envío y finalmente envía los datos

## Programación de la aplicación

### Pantalla home\_screen.dart

import 'package:app\_abejas/screens/detalles\_screen.dart';

import 'package:flutter/material.dart';

import 'package:google\_maps\_flutter/google\_maps\_flutter.dart';

import 'package:fl\_chart/fl\_chart.dart';

Importa las librerías necesarias para la aplicación y otras pantallas asociadas, entre las librerías están componentes visuales de Flutter, mapa de Google y una librería de para gráficos

class Sensor {

  final String nombre;

  final LatLng ubicacion;

  final List<double> temperaturaExt;

  final List<double> temperaturaInt;

  final List<double> humedadExt;

  final List<double> humedadInt;

  final Color color;

  Sensor({

    required this.nombre,

    required this.ubicacion,

    required this.temperaturaExt,

    required this.temperaturaInt,

    required this.humedadExt,

    required this.humedadInt,

    required this.color,

  });

}

Esta clase simula la obtención de datos desde la base de datos para testeo de la aplicación

class HomeScreen extends StatefulWidget {

  const HomeScreen({Key? key}) : super(key: key);

  @override

  State<HomeScreen> createState() => \_HomeScreenState();

}

Crea una instancia de la pantalla principal

class \_HomeScreenState extends State<HomeScreen> {

  late GoogleMapController mapController;

  Sensor? sensorSeleccionado;

Clase que controla el estado de la pantalla

final List<Sensor> sensores = [

    Sensor(

      nombre: 'Sensor 1',

      ubicacion: LatLng(-32.901757, -71.227288),

      temperaturaExt: [25, 26, 27, 28],

      temperaturaInt: [35, 36, 37, 38],

      humedadExt: [50, 55, 54, 57],

      humedadInt: [70, 70, 74, 75],

      color: Colors.red,

    ),

    Sensor(

      nombre: 'Sensor 2',

      ubicacion: LatLng(-32.901640, -71.226836),

      temperaturaExt: [30, 28, 29, 31],

      temperaturaInt: [35, 36, 37, 38],

      humedadExt: [50, 55, 54, 57],

      humedadInt: [70, 70, 74, 75],

      color: Colors.green,

    ),

    Sensor(

      nombre: 'Sensor 3',

      ubicacion: LatLng(-32.902047, -71.227144),

      temperaturaExt: [25, 26, 24, 27],

      temperaturaInt: [30, 32, 33, 31],

      humedadExt: [50, 55, 54, 57],

      humedadInt: [70, 70, 74, 75],

      color: Colors.blue,

    ),

  ];

Lista de sensores con datos simulados, estos datos son exactamente el mismo formato que los dispositivos envían a los servicios de AWS

Set<Marker> get \_crearMarcadores {

    return sensores.map((sensor) {

      return Marker(

        markerId: MarkerId(sensor.nombre),

        position: sensor.ubicacion,

        infoWindow: InfoWindow(title: sensor.nombre),

        icon: BitmapDescriptor.defaultMarkerWithHue(\_colorToHue(sensor.color)),

      );

    }).toSet();

  }

Método para crear los marcadores en el mapa a partir de la lista de sensores

double \_colorToHue(Color color) {

    if (color == Colors.red) return BitmapDescriptor.hueRed;

    if (color == Colors.green) return BitmapDescriptor.hueGreen;

    if (color == Colors.blue) return BitmapDescriptor.hueBlue;

    return BitmapDescriptor.hueAzure;

  }

Método para convertir un color a un valor hue que usa Google maps

@override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: const Text('Mapa Sensores')),

      body: Column(

        children: [

          Expanded(

            flex: 1,

            child: GoogleMap(

              mapType: MapType.satellite,

              initialCameraPosition: CameraPosition(

                target: sensores[0].ubicacion,

                zoom: 16,

              ),

              markers: \_crearMarcadores,

              onMapCreated: (controller) {

                mapController = controller;

              },

            ),

          ),

Método que construye la interfaz gráfica, construye en la parte superior una instancia de mapa de Google, con el estilo de satélite, zoom inicial de 16, muestra los marcadores y guarda el controlador del mapa

Expanded(

            flex: 1,

            child: Column(

              children: [

                Expanded(

                  child: ListView.builder(

                    itemCount: sensores.length,

                    itemBuilder: (context, index) {

                      final sensor = sensores[index];

                      return ListTile(

                        leading: Icon(Icons.location\_on, color: sensor.color),

                        title: Text(sensor.nombre),

                        trailing: IconButton(

                          icon: Icon(Icons.arrow\_forward),

                          onPressed: () {

                            Navigator.push(

                              context,

                              MaterialPageRoute(

                                builder: (context) => detalles\_screen(sensor: sensor),

                              ),

                            );

                          },

                        ),

                        onTap: () {

                          setState(() {

                            sensorSeleccionado = sensor;

                          });

                          mapController.animateCamera(

                            CameraUpdate.newLatLngZoom(sensor.ubicacion, 19),

                          );

                        },

                      );

                    },

                  ),

                ),

En la parte inferior de la pantalla muestra una lista de sensores y el grafico asociado, recorre la lista de sensores, también en la lista de los sensores se genera un botón que lleva a la pantalla de detalles, al tocar el sensor el mapa lleva a la dirección del sensor haciendo un zoom de 19

 if (sensorSeleccionado != null) ...[

                  Padding(

                    padding: const EdgeInsets.all(8.0),

                    child: Text('temperaturaExt de ${sensorSeleccionado!.nombre}'),

                  ),

                  SizedBox(

                    height: 200,

                    child: LineChart(

                      LineChartData(

                        gridData: FlGridData(show: false),

                        titlesData: FlTitlesData(show: true),

                        borderData: FlBorderData(show: true),

                        lineBarsData: [

                          LineChartBarData(

                            spots: sensorSeleccionado!.temperaturaExt

                                .asMap()

                                .map((index, value) {

                                  return MapEntry(index, FlSpot(index.toDouble(), value));

                                })

                                .values

                                .toList()

                          ),

                        ],

                      ),

                    ),

                  ),

                ],

              ],

            ),

          ),

        ],

      ),

    );

  }

}

Si hay un sensor seleccionado se muestra el grafico de las mediciones de ese sensor

### Pantalla detalles\_screen.dart

import 'package:flutter/material.dart';

import 'package:fl\_chart/fl\_chart.dart';

import 'home\_screen.dart';

Se importan las librerías necesarias, fl\_chart permite la creación de gráficos

class detalles\_screen extends StatefulWidget {

  final Sensor sensor;

  const detalles\_screen({Key? key, required this.sensor}) : super(key: key);

  @override

  \_detalles\_screenState createState() => \_detalles\_screenState();

}

Se crea un statefulwidget para mostrar los detalles de un sensor especifico que contiene el constructor que recibe el sensor

class \_detalles\_screenState extends State<detalles\_screen> with SingleTickerProviderStateMixin {

  late TabController \_tabController;

  @override

  void initState() {

    super.initState();

    \_tabController = TabController(length: 4, vsync: this);

  }

  @override

  void dispose() {

    \_tabController.dispose();

    super.dispose();

  }

Esta clase maneja el estado de la pantalla donde se inicializa el tab controller con cuatro pestañas (una para cada variable del sensor) y se liberan los recursos cuando se cierra esta pantalla

@override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Detalle de ${widget.sensor.nombre}'),

        bottom: TabBar(

          controller: \_tabController,

          tabs: const [

            Tab(text: 'Temp. Ext'),

            Tab(text: 'Temp. Int'),

            Tab(text: 'Humedad Ext'),

            Tab(text: 'Humedad Int'),

          ],

        ),

      ),

      body: TabBarView(

        controller: \_tabController,

        children: [

          \_buildChartAndTable(widget.sensor.temperaturaExt, 'Temperatura Externa'),

          \_buildChartAndTable(widget.sensor.temperaturaInt, 'Temperatura Interna'),

          \_buildChartAndTable(widget.sensor.humedadExt, 'Humedad Externa'),

          \_buildChartAndTable(widget.sensor.humedadInt, 'Humedad Interna'),

        ],

      ),

    );

  }

Appbar con título dinámico según el sensor seleccionado, donde el tabcontroller indica en que pestaña esta y luego muestra el contenido de la pestaña, en este caso, corresponde a las humedades y temperaturas internas y externas

 Widget \_buildChartAndTable(List<double> data, String label) {

    return Padding(

      padding: const EdgeInsets.all(12.0),

      child: Column(

        children: [

          // Gráfico

          SizedBox(

            height: 200,

            child: LineChart(

              LineChartData(

                lineBarsData: [

                  LineChartBarData(

                    spots: data.asMap().entries.map(

                      (e) => FlSpot(e.key.toDouble(), e.value),

                    ).toList(),

                    isCurved: true,

                    dotData: FlDotData(show: true),

                  ),

                ],

              ),

            ),

          ),

          const SizedBox(height: 20),

          // Título de la variable

          Text(label, style: Theme.of(context).textTheme.headlineMedium),

          const SizedBox(height: 10),

          // Tabla de mediciones

          Expanded(

            child: ListView.builder(

              itemCount: data.length,

              itemBuilder: (context, index) {

                return ListTile(

                  leading: Text('#${index + 1}'),

                  title: Text('${data[index]}'),

                );

              },

            ),

          ),

        ],

      ),

    );

  }

}

Este método crea un gráfico y una tabla para mostrar los datos del sensor seleccionado

## Configuración de AWS para recibir datos de los sensores

Para configurar AWS IoT Core para recibir datos por MQTT se necesita crear una cuenta de AWS

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Ilustración 7: Pantalla AWS IoT Core

Se debe ir a la sección All devices -> Things -> Create things -> Create single thing para poder crear un nuevo dispositivo que conectaremos por MQTT

Luego de seleccionar un nombre para nuestro thing y de seleccionar autogenerar los certificados se nos presenta la opción de crear una policy que nos permite garantizar acceso a nuestro thing de las funciones de AWS

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Ilustración18: Policy seleccionada en mi thing

Agregando este script JSON damos acceso al thing todas las funciones de IoT disponibles, luego es necesario descargar el certificado del dispositivo, la private key y el Amazon trust services endpoint 1, estos tres documentos tienen que ser abiertos en un editor de texto que nos permita copiar el contenido y agregarlo a secrets.h del dispositivo que le permitirá conectarse a las funciones de MQTT

Antes de continuar con la configuración, se necesita entrar a las configuraciones de cuenta de AWS en la parte inferior izquierda del panel de control Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 19: Settings

Donde tendremos que dirigirnos a la sección de view domain configurations y copiar el domain name para poder pegarlo en el secrets.h

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración 20: Domain name

### secrets.h

#include <pgmspace.h>

#define SECRET

const char WIFI\_SSID[] = "Farias";

const char WIFI\_PASSWORD[] = "ssbb1997!";

#define THINGNAME "sensor\_colmena\_01"

int8\_t TIME\_ZONE = -4;

const char MQTT\_HOST[] = "a3n8687rz6kzhp-ats.iot.us-east-2.amazonaws.com";

Se definen las variables a ocupar en el dispositivo, entre estas están el SSID de la red wifi, la contraseña, el nombre del thing que creamos, la zona horaria y el domain name que copiamos desde configuración

static const char cacert[] PROGMEM = R"EOF(

)EOF";

// Copy contents from XXXXXXXX-certificate.pem.crt here ▼

static const char client\_cert[] PROGMEM = R"KEY(

)KEY";

// Copy contents from  XXXXXXXX-private.pem.key here ▼

static const char privkey[] PROGMEM = R"KEY(

)KEY";

Finalmente se tiene que pegar los certificados y keys descargados en la creación del thing, en este punto el dispositivo debería estar publicando datos en el topic seleccionado para publicación de los datos dentro de la sección MQTT test client

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Ilustración 21: MQTT test client

### Creación de Rules para comunicación con Lambda

Para guardar los datos enviados al topic de MQTT en la database de DynamoDB se necesita crear una regla que mande esta información a Lambda, la cual esencialmente es un traductor que permite comunicación entre variados servicios de AWS

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración 22: Sección Rules

Seleccionar la creación de Rules, insertar un nombre para la rule

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 23: SQL statement

Luego introducir un statement SQL que nos de acceso a todo el contenido de cada mensaje enviado a el pub seleccionado en el código del dispositivo hay que seleccionar la rule action de lambda, esto nos dará la opción de crear una nueva función de lambda

### Función Lambda

import boto3

def lambda\_handler(event, context):

client = boto3.client('dynamodb')

response = client.put\_item(

TableName='sensor\_DynamoDB\_Table',

Item={

'fechaHora': {'S': event['fechaHora']},

'sensorId': {'S': event['sensorId']},

'humExt': {'N': str(event['humExt'])},

'humInt': {'N': str(event['humInt'])},

'tempExt': {'N': str(event['tempExt'])},

'tempInt': {'N': str(event['tempInt'])},

'lng': {'N': str(event['lng'])},

'lat': {'N': str(event['lat'])}

}

)

Con esta función de lambda creada usando Python 3.9, podemos dividir el mensaje enviado por el dispositivo en variables numéricas o strings denominadas por N o S, como se puede ver también existe una variable llamada sensor\_DynamoDB\_Table, que es la tabla donde se guardara nuestra data recién dividida

### Crear DynamoDB table

Para crear la tabla de DynamoDB se necesita ingresar a dicho portal y crear una tabla nueva con el nombre estipulado en el script de lambda y seleccionar una variable para crear una llave de partición, en este caso, estoy utilizando la fechaHora como llave

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Ilustración 4: Creación de table

### Crear un Role para lambda a DynamoDB

Para esto necesitamos dirigirnos al panel principal de IAM -> Roles -> Create Role, seleccionar lambda y dar fullaccess a DynamoDB

Captura de pantalla con letras y números

El contenido generado por IA puede ser incorrecto.

Ilustración 25: Configuration -> Permissions

Finalmente volver a nuestra función de Lambda y dirigirse al panel de configuración-> Permissions donde podremos agregar el role recién hecho a nuestra función, esto le dará acceso para agregar contenido a nuestra tabla de DynamoDB cuando un mensaje nuevo es agregado al topic seleccionado

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Ilustración 26: Tabla de DynamoDB siendo poblada por data del sensor

Posteriormente se puede crear otra función de lambda para enviar la data de esta tabla a la aplicación de Flutter

# Riesgos gatillados

|  |  |  |
| --- | --- | --- |
| Riesgo Gatillado | Descripción | Acción Tomada |
| Problemas con placas ESP32 | Algunas placas ESP32 presentaron problemas al momento de cargar código, lo que detuvo parte del desarrollo de hardware. | Se optó por utilizar placas D1 Mini (ESP8266) para continuar con las pruebas y el envío de datos. |
| Desconocimiento de AWS | No tenía conocimientos previos sobre la plataforma AWS ni su sistema de comunicación IoT. | Se realizaron cursos, documentación y pruebas para lograr la conexión mediante MQTT y gestión de dispositivos. |
| Desconocimiento de Flutter | No manejaba Flutter para el desarrollo de la aplicación móvil. | Se aprendió la herramienta mediante documentación oficial, tutoriales y prácticas, logrando desarrollar parte importante de la app. |
| Tiempo limitado | Dificultad para avanzar debido a la espera de la entrega de compra de materiales | Organización de tareas y extensión de horarios de trabajo en las noches para avanzar en el proyecto. |
| Problemas iniciales con MQTT | Se presentaron fallas en la transmisión de datos por mala conexión WiFi. | Se implementó reconexión automática y manejo de errores para asegurar la entrega de datos. |

# Ejecución de pruebas

Durante el desarrollo del proyecto se realizaron distintas pruebas tanto en la parte de hardware (sensores y placas) como en la aplicación móvil. Estas pruebas permitieron validar el funcionamiento básico del sistema y detectar errores a corregir.

## Pruebas Realizadas:

### Prueba de Sensores

* Sensores utilizados: DHT22 y SHT41.
* Objetivo: Verificar lectura de temperatura y humedad interior/exterior de los cajones (actualmente usando mi pc como referencia a un cajón).
* Resultado:
  + El sensor SHT41 entregó mejor estabilidad y precisión.
  + El sensor DHT22 presentó variaciones importantes en exteriores.

### Prueba de App Móvil

* Objetivo: Verificar visualización de datos, diseño de pantallas y funcionamiento de mapas y gráficos.
* Resultado:
  + Pantallas diseñadas correctamente.
  + Gráficos funcionales mostrando datos de prueba.
  + Función de mapa operativa (aún sin datos reales de GPS).

### Prueba de Integración Parcial

* Objetivo: Verificar funcionamiento básico del sistema de almacenamiento
* Resultado:
  + Pruebas exitosas con datos reales enviados desde sensores a la base de datos y visualizados mediante herramientas de prueba (aún falta integración directa a la app móvil).

# Evidencia de monitoreo y control del Sprint

Para la planificación, gestión y control de las tareas del proyecto se utilizó la plataforma Jira, en la cual se definieron las historias de usuario, se crearon los sprints y se realizó un seguimiento del avance de las tareas.

A continuación, se presentan capturas de evidencia de dicha gestión:

Interfaz de usuario gráfica, Tabla

El contenido generado por IA puede ser incorrecto.

Ilustración 27: Sprint 1 terminado, con historias de usuario asociadas

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Ilustración 28: Sprint 2 y Sprint 3 listos para su inicio en las fechas correspondientes

# Evidencia de liberación MVP

Actualmente se cuenta con un Producto Mínimo Viable (MVP) funcional que permite cumplir con alguno de los objetivos básicos del proyecto.

## Hardware:

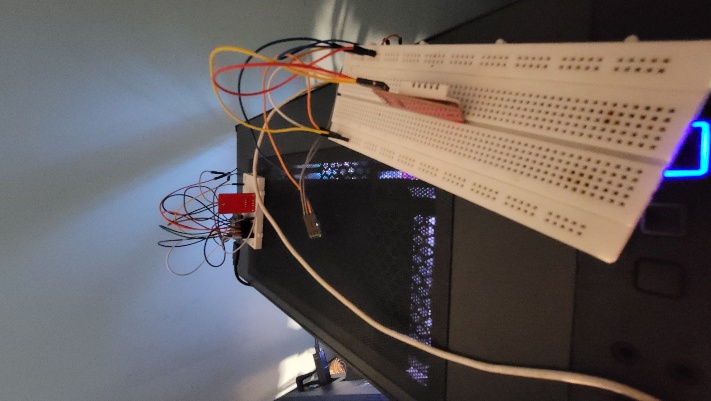


Ilustración 29: Sensor1 y Sensor2 enviando mediciones desde afuera y dentro de mi computador

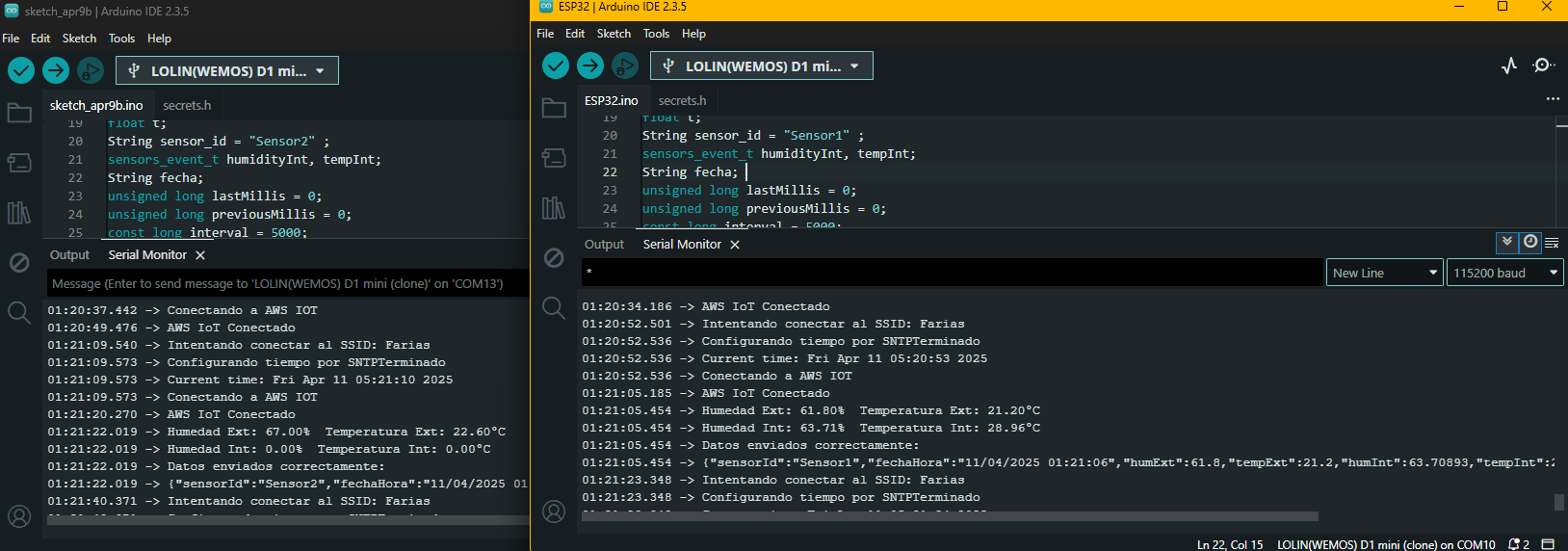


Ilustración 30: Serial monitor del Sensor1 y Sensor2 mandando data a AWS IoT Core

## DynamoDB:

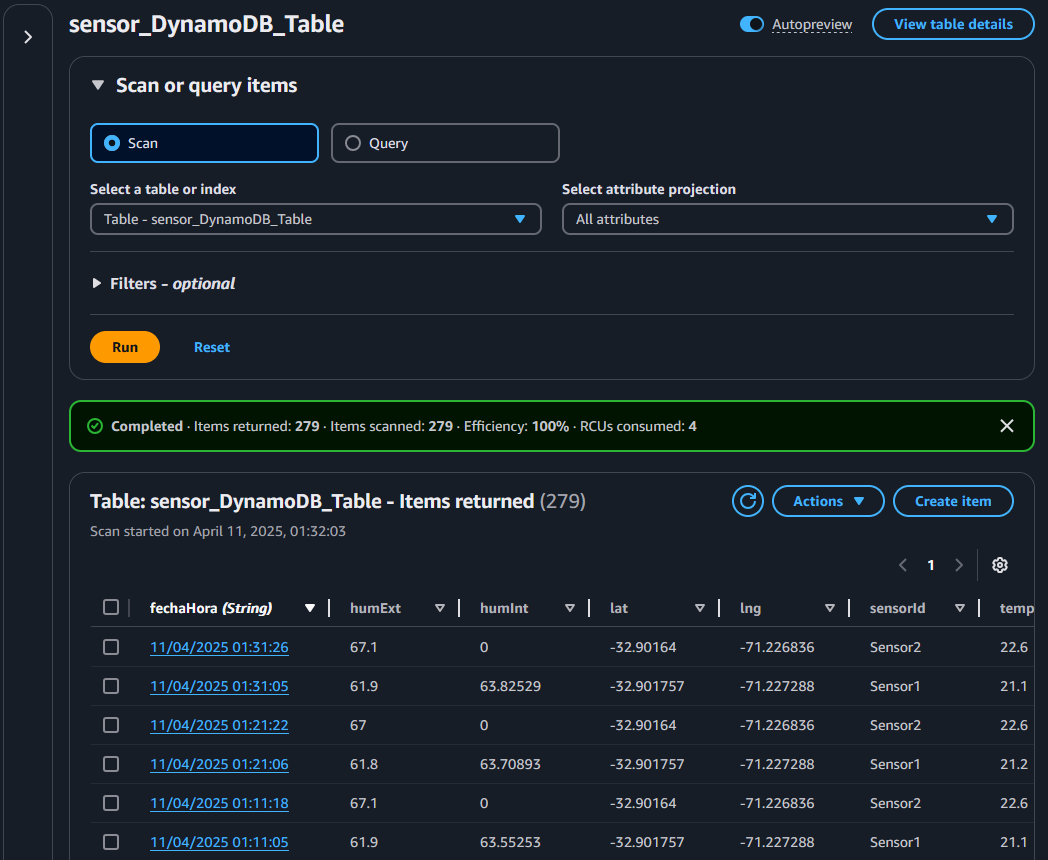


Ilustración 31: Tabla en DynamoDB donde están almacenadas las mediciones de los sensores

## Aplicación móvil:

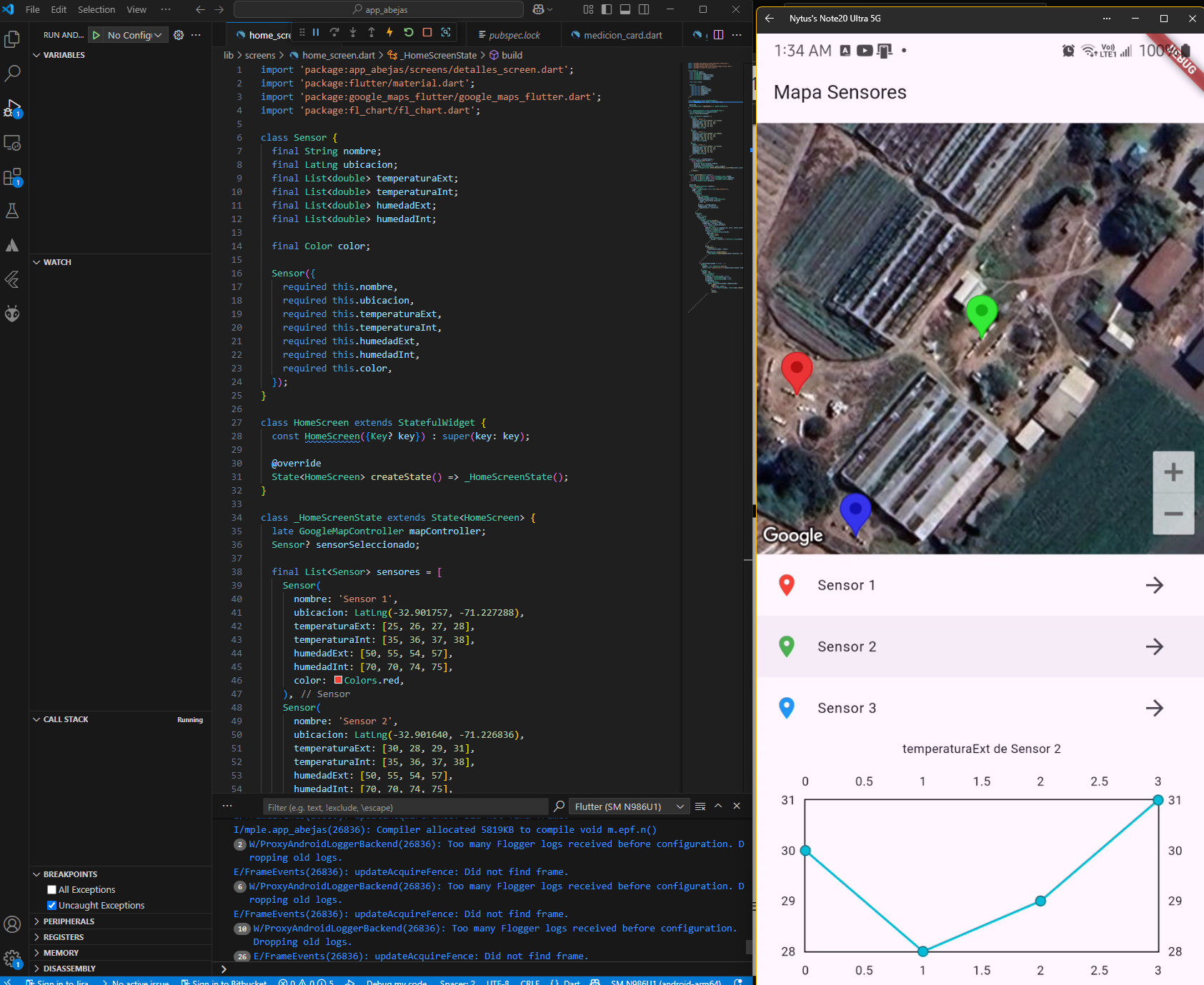


Ilustración 2: transmisión de la app funcionando desde mi teléfono

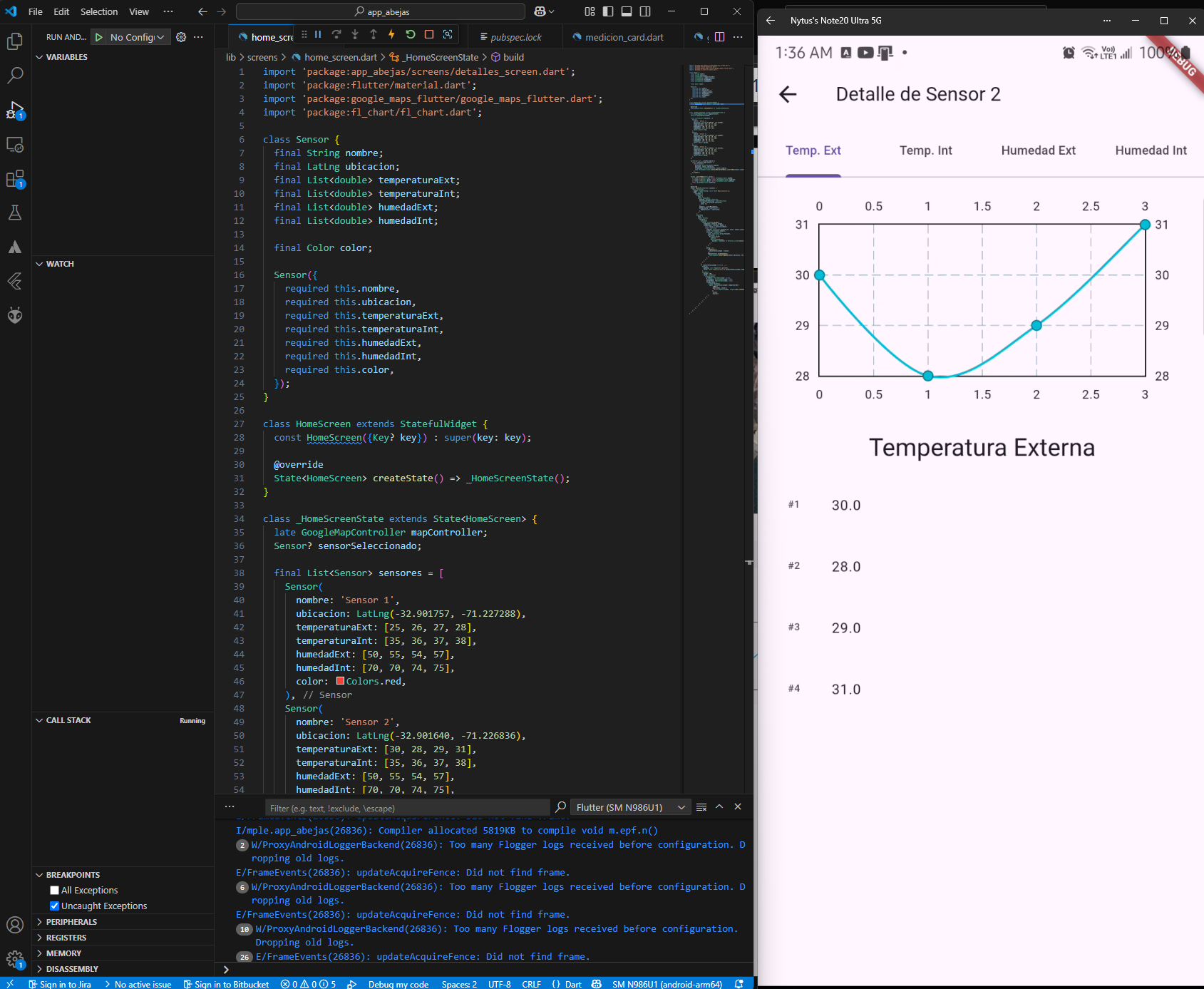


Ilustración : transmisión de los detalles\_screen funcionando desde mi teléfono

# Problemas Abiertos

Problemas Técnicos:

* Integración final de la aplicación móvil con la base de datos (DynamoDB).
* Incorporar visualización en tiempo real de datos recibidos desde los sensores en la app.
* Comprar nuevas placas de desarrollo que permitan la incorporación de los módulos de red móvil (ESP32)

Problemas de Conocimiento:

* Necesidad de profundizar conocimientos en AWS para una mejor gestión de los dispositivos y la base de datos.
* Aprendizaje de buenas prácticas en Flutter y arquitectura de apps móviles para un mejor desarrollo de la aplicación.

Tareas Pendientes:

* Despliegue de la app en un APK instalable.
* Realizar pruebas de campo en un ambiente real.
* Ajustes en la interfaz gráfica de usuario.
* Documentación final del proyecto.

# Bibliografía

España, G. (11 de abril de 2022). Abejas - ES | Greenpeace España. Obtenido de ES | Greenpeace España: https://es.greenpeace.org/es/trabajamos-en/agricultura/abejas/

Higes, M. (Octubre de 2010). SAG. Obtenido de SAG: https://www.sag.gob.cl/sites/default/files/informe\_caso\_nosema\_ceranae\_biobio\_2009.pdf

Langowska, A., Zawilak, M., Sparks, T., Glazaczow, A., Tomkins, P., & Tryjanowski, P. (24 de diciembre de 2016). Long-term effect of temperature on honey yield and honeybee phenology. International Journal of Biometeorology, 61(6), págs. 1125 - 1132. doi:10.1007/s00484-016-1293-x

Latam, M. (. (20 de mayo de 2021). Las abejas en Chile están en serio peligro y es culpa de sus habitantes. Obtenido de BioBioChile - La Red de Prensa Más Grande de Chile: https://www.biobiochile.cl/especial/aqui-tierra/noticias/2021/05/20/dia-mundial-de-las-abejas-cientificos-destacan-la-urgencia-de-transitar-hacia-la-agroecologia.shtml

Tapia-González, J., Alcazar-Oceguera, G., Macías-Macías, J., Contreras-Escareño, F., Tapia-Rivera, J., Chavoya-Moreno, F., & Martínez-González, J. (28 de junio de 2017). Nosemosis in worker bees and their relationship with environmental factors in Jalisco, Mexico. Revista Mexicana de Ciencias Pecuarias.

Today, E. (. (25 de mayo de 2021). This old bee house: Study deems hive boxes drafty, inefficient. Obtenido de Entomology Today: https://entomologytoday.org/2021/04/16/honey-bee-hive-boxes-drafty-inefficient-temperature/

España, G. (11 de abril de 2022). Abejas - ES | Greenpeace España. Obtenido de ES | Greenpeace España: <https://es.greenpeace.org/es/trabajamos-en/agricultura/abejas/>

(ThingsBoard Cloud, n.d.) – Plataforma Thingsboard

(Open Weather Map, n.d.) – Api OpenWeatherMap

(Fritzing, s.f.) – Modelador de dispositivos prototipo

<https://www.3bee.com/es/tecnologia/>

https://bibliotecadigital.fia.cl/entities/proyecto/54d63235-795b-4c94-9f52-8a385ddc9d04/full