# TAMAPLANT

NOMBRE: Jonattan Larrea | Nelson Aravena

CARRERA: Ingeniería en Informática

ASIGNATURA: Proyecto de Título

PROFESOR: Javier Miles

**CONTENIDO**

[TAMAPLANT 1](#_Toc213763257)

[1 INTRODUCCIÓN 6](#_Toc213763258)

[2 TECNOLOGÍAS A IMPLEMENTAR 6](#_Toc213763259)

[2.1 COMPARACIÓN DE TECNOLOGÍAS 6](#_Toc213763260)

[2.1.1 Tecnología Hardware: Todo lo correspondiente a tecnología de carácter físico.  IoT 6](#_Toc213763261)

[2.1.2 Tecnología Software: Todo lo correspondiente a desarrollo. 16](#_Toc213763262)

[3 ANÁLISIS COMPARATIVO CUALITATIVO (IOT / HARDWARE) 24](#_Toc213763263)

[3.1 Placas programables 24](#_Toc213763264)

[3.2 Módulos / Sensores de Temperatura 24](#_Toc213763265)

[3.3 Módulos / Sensores de Humedad 24](#_Toc213763266)

[3.4 Módulos / Sensores Barométricos (Presión atmosférica) 25](#_Toc213763267)

[3.5 Módulos / Sensores de luminosidad 25](#_Toc213763268)

[4 ANÁLISIS COMPARATIVO CUALITATIVO (SOFTWARE) 26](#_Toc213763269)

[4.1 Bases de datos 26](#_Toc213763270)

[4.2 BackEnd 26](#_Toc213763271)

[4.3 FrontEnd 27](#_Toc213763272)

[4.3.1 Comunicación y servicios 27](#_Toc213763273)

[4.3.2 Métodos de Transmisión y visualización de datos 27](#_Toc213763274)

[5 ANÁLISIS COMPARATIVO CUANTITATIVO 29](#_Toc213763275)

[5.1 PLACAS PROGRAMABLES 29](#_Toc213763276)

[5.2 Módulos / Sensores de Temperatura 30](#_Toc213763277)

[5.3 Módulos / Sensores de humedad 30](#_Toc213763278)

[5.4 Módulos / Sensores Barométricos 31](#_Toc213763279)

[5.5 Módulos / Luminosidad 31](#_Toc213763280)

[5.6 Métrica General Estándar (MGE) para comparación de Bases de Datos 32](#_Toc213763281)

[5.7 Métodos de Transmisión y visualización de datos 33](#_Toc213763282)

[5.8 Servidor virtual (VPS) 34](#_Toc213763283)

[5.9 Servidor Cloud 35](#_Toc213763284)

[5.10 PLACAS CUANTITATIVO 35](#_Toc213763285)

[5.10.1 Consumo de Energía (mA) 35](#_Toc213763286)

[5.10.2 Rendimiento / CPU 36](#_Toc213763287)

[5.10.3 Almacenamiento / Memoria 36](#_Toc213763288)

[5.11 SENSORES CUANTITATIVO (SENSOR DE TEMPERATURA) 36](#_Toc213763289)

[5.11.1 Consumo de Energía (mA) 36](#_Toc213763290)

[5.11.2 Rendimiento (CPU) 37](#_Toc213763291)

[5.12 SENSORES CUANTITATIVO (SENSOR DE HUMEDAD) 37](#_Toc213763292)

[5.12.1 Consumo de Energía (mA) 37](#_Toc213763293)

[5.12.2 Rendimiento (CPU) 37](#_Toc213763294)

[5.13 SENSORES CUANTITATIVO (SENSOR DE BAROMETRICIDAD) 37](#_Toc213763295)

[5.13.1 Consumo de Energía (mA) 37](#_Toc213763296)

[5.13.2 Rendimiento (CPU) 38](#_Toc213763297)

[5.14 SENSORES CUANTITATIVO (SENSOR DE LUMINOSIDAD) 38](#_Toc213763298)

[5.14.1 Consumo de Energía (mA) 38](#_Toc213763299)

[5.14.2 Rendimiento (CPU) 38](#_Toc213763300)

[6 Selección de tecnologías (IOT / Hardware) 39](#_Toc213763301)

[6.1 Factibilidad Técnica: Se analizan precisión, compatibilidad y facilidad de integración de los sensores y la placa seleccionada 39](#_Toc213763302)

[6.1.1 Conclusión factibilidad técnica: 39](#_Toc213763303)

[6.2 Factibilidad Económica: 40](#_Toc213763304)

[6.2.1 Conclusión factibilidad económica: 40](#_Toc213763305)

[6.3 Factibilidad Implementativa 40](#_Toc213763306)

[6.3.1 Conclusión implementativa: 41](#_Toc213763307)

[**6.4** Selección de tecnologías (Software) 41](#_Toc213763308)

[6.4.1 Factibilidad Técnica 41](#_Toc213763309)

[6.4.2 Factibilidad Operativa 42](#_Toc213763310)

[6.5 Factibilidad Implementativa 42](#_Toc213763311)

[7 DETALLE DE ARQUITECTURA A IMPLEMENTAR 43](#_Toc213763312)

[7.1 Diagrama BPMN 43](#_Toc213763313)

[7.2 Diagrama de Casos de Uso 44](#_Toc213763314)

[7.3 Diagrama de Componentes 44](#_Toc213763315)

[7.4 Modelo de Datos 45](#_Toc213763316)

[7.5 Topología de Comunicación 45](#_Toc213763317)

[45](#_Toc213763318)

[7.6 Topología de Infraestructura 46](#_Toc213763319)

[7.7 Topología de Arquitectura 47](#_Toc213763320)

[8 IMPLEMENTACIÓN DE LOS KPI Y SLA 47](#_Toc213763321)

[8.1 Descripción de los KPI 47](#_Toc213763322)

[8.2 Descripción de SLA 48](#_Toc213763323)

[9 PLAN DE PRUEBAS Y ASEGURAMIENTO DE CALIDAD 49](#_Toc213763324)

[9.1 Plan de Pruebas 49](#_Toc213763325)

[9.2 Normas y Estándares 49](#_Toc213763326)

[10 PLAN DE IMPLEMENTACIÓN 50](#_Toc213763327)

[10.1 Gestión de Disponibilidad 50](#_Toc213763328)

[10.2 Gestión de Continuidad 51](#_Toc213763329)

[10.2.1 Política de Respaldo 51](#_Toc213763330)

[10.2.2 Restauración del Servicio 51](#_Toc213763331)

[10.2.3 Código Fuente 51](#_Toc213763332)

[10.3 Plan de Mantención 51](#_Toc213763333)

[10.3.1 Mantenimiento Preventivo de Software 51](#_Toc213763334)

[10.3.2 Mantenimiento Correctivo 51](#_Toc213763335)

[10.3.3 Mantenimiento Evolutivo 51](#_Toc213763336)

[11 CONCLUSIONES 51](#_Toc213763337)

[REFERENCIAS BIBLIOGRÁFICAS 52](#_Toc213763338)

# INTRODUCCIÓN

El presente informe se centra en el desarrollo de un Gemelo Digital de una planta, con el objetivo de recopilar, analizar y monitorear datos provenientes de una red de sensores IoT. Estos sensores miden variables ambientales se vital importancia como la humedad, el pH, la temperatura, la luminosidad y los niveles de CO2.

El propósito principal es transformar el seguimiento de las condiciones de una planta, que actualmente se realiza de manera manual, en un proceso digitalizado, centralizado, en tiempo real y proactivo. Esta plataforma permitirá a los usuarios visualizar de manera inmersiva el estado de la planta, identificar alteraciones y anticiparse a posibles problemas. El análisis de los datos recolectados mediante el Gemelo Digital creará la base para una futura automatización de los procesos, como el riego, optimizando los recursos y promoviendo la sostenibilidad.

# TECNOLOGÍAS A IMPLEMENTAR

## COMPARACIÓN DE TECNOLOGÍAS

### Tecnología Hardware: Todo lo correspondiente a tecnología de carácter físico.  IoT

#### Placas:

* + Arduino uno:

Un circuito electrónico

El contenido generado por IA puede ser incorrecto.

* **Descripción**: Placa clásica basada en microcontrolador ATmega328P, ideal para proyectos básicos de electrónica.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Gran comunidad y soporte | No tiene conectividad WiFi |
| Muy fácil de usar | Poca memoria y procesamiento limitado |

* Arduino uno leonardo:



* **Descripción**: Similar al UNO, pero basado en ATmega32u4. Permite emular dispositivos USB como teclado o ratón.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Puede comportarse como dispositivo USB | Tampoco tiene WiFi integrado |
| Buen soporte con sensores comunes | Compatibilidad menor con algunos shields |

* ESP32;

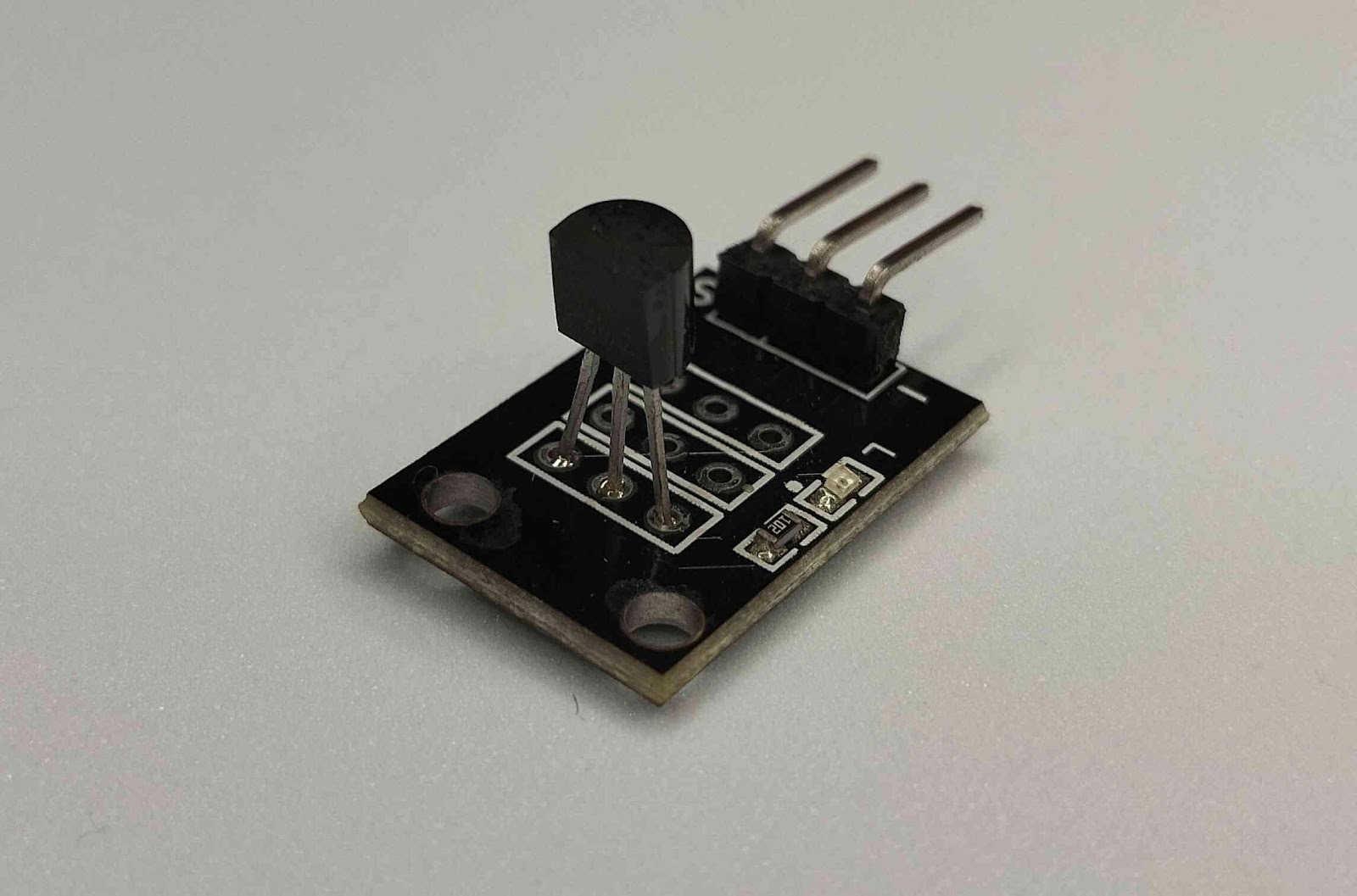


* **Descripción**: Microcontrolador de 32 bits con WiFi y Bluetooth integrados, más potente y moderno.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Conectividad integrada (WiFi + BT) | Requiere más cuidado con niveles de voltaje |
| Alta velocidad y más memoria | Curva de aprendizaje un poco mayor que Arduino |

#### Sensor de temperatura:

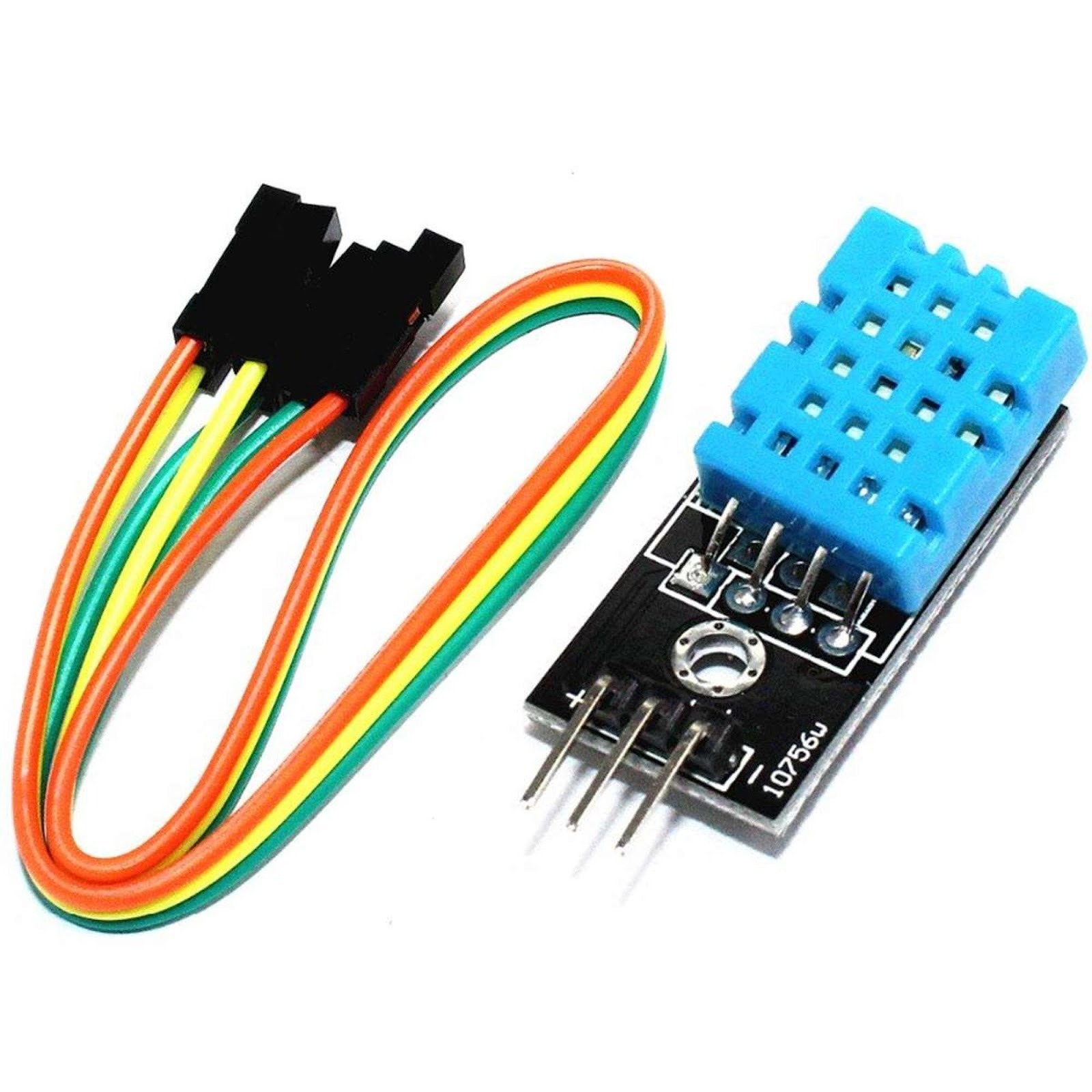
* + Sensor de temperatura Módulo KY-001



* **Descripción**: Sensor digital de temperatura de alta precisión, usa protocolo OneWire.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Precisión (+0.6°C) | Requiere librerías específicas |
| Permite conectar varios sensores al mismo pin | Cableado ligeramente más complejo |

* Sensor de temperatura DHT11



* **Descripción**: Sensor digital que mide temperatura y humedad, ideal para proyectos educativos.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Fácil de usar | Baja precisión (+- 2°C) |
| Incluye lectura de humedad | Lento en actualización |

* Modulo sensor de temperatura DS18B20

A black and red cable

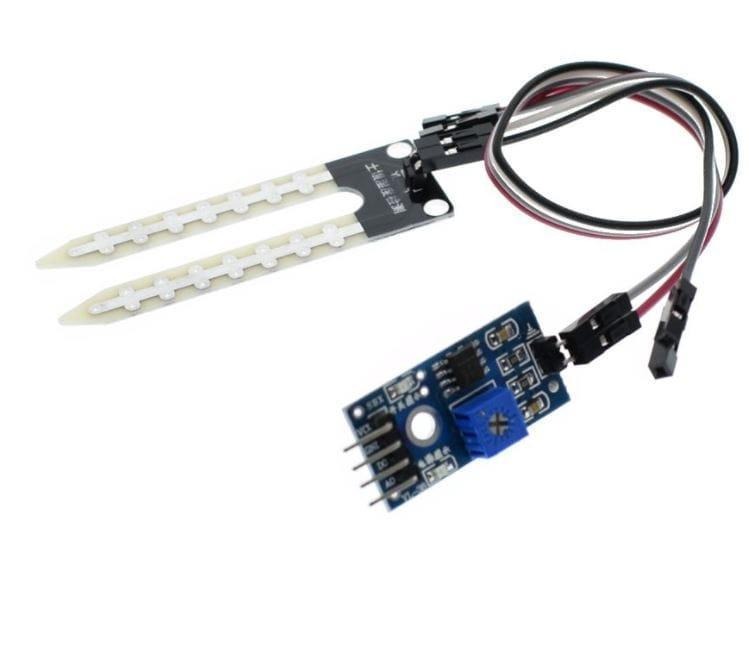
AI-generated content may be incorrect.

* **Descripción**: Sensor digital de temperatura sumergible que mide con precisión desde -55°C hasta + 125°C.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Alta precisión y fácil lectura por su salida digital | Solo mide temperatura, no otros parámetros ambientales |
| Puede ser sumergible, ideal para medir temperatura en líquidos o ambientes húmedos. | La comunicación 1-wire puede ser más compleja de implementar. |

#### Sensores de Humedad:

* Modulo sensor de humedad FC-28 (HIGRÓMETRO)



* **Descripción**: Sensor de humedad que mide la conductividad eléctrica entre dos varillas metálicas.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Muy económico | Se corroe con el tiempo |
| Fácil de usar | Baja precisión y vida útil |

* Sensor Capacitivo de Humedad del suelo



* **Descripción**: Sensor que mide humedad a través de la variación de capacitancia del sustrato

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Más preciso y estable | Ligeramente más caro |
| No se corroe |  |

* Sensor de presión atmosférica BMP280

A purple circuit board with white text and yellow circles

AI-generated content may be incorrect.

* **Descripción**:  Sensor digital que mide presión atmosférica y temperatura, ideal para proyectos de meteorología o altitud.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Alta precisión en las mediciones | No mide humedad, a diferencia de otros sensores similares. |
| Bajo consumo de energía, ideal para dispositivos portátiles. | Es sensible a cambios bruscos de temperatura, lo que puede afectar la precisión momentáneamente. |

#### Sensores de presión

* Sensor de presión atmosférica BME 280

A purple rectangular object with many small black and white objects

AI-generated content may be incorrect.

* **Descripción**: Sensor digital que mide temperatura, humedad y presión atmosférica, muy usado en proyectos de monitoreo ambiental.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Mide tres variables con un solo dispositivo | Es más caro que sensores que solo miden una variable. |
| Es preciso y estable incluso en diferentes condiciones ambientales. | Requiere una configuración inicial más compleja para obtener todos los datos correctamente |

* Sensor de presión atmosférica BMP180

A blue circuit board with white text

AI-generated content may be incorrect.

* **Descripción**: Sensor digital que mide presión atmosférica y temperatura, común en proyectos de altitud y clima.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Es económico y confiable para medir presión y temperatura. | No mide humedad, a diferencia de sensores más nuevos como el BME 280 |
| Tiene bajo consumo de energía, ideal para sistemas portátiles | Es un modelo antiguo, reemplazado por versiones más precisas como el BMP 280 |

#### Sensores de Luminosidad:

* Sensor de luminosidad (Lux) BH1750

**A blue circuit board with several small pins

AI-generated content may be incorrect.**

* **Descripción:** Sensor digital de luminosidad que mide la intensidad de luz en lux, similar a cómo la percibe el ojo humano.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Entrega lecturas precisas y directas en lux, sin cálculos adicionales. | No funciona bien con fuentes de luz infrarrojas o no visibles. |
| Tiene bajo consumo de energía y comunicaciones sencillas mediante I2C. | Puede ser sensible a la orientación o al polvo, afectando las mediciones. |

* Sensor de luminosidad Fotoresistivo / fotorresistencia

A blue circuit board with a blue square and red round object

AI-generated content may be incorrect.

* **Descripción**: sensor fotoresistivo que mide la intensidad de luz mediante una resistencia que varía según la cantidad de luz recibida.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Es muy económico y fácil de usar. | Su precisión es baja comparada con sensores digitales como el BH1750. |
| Permite una respuesta rápida a los cambios de iluminación. | Puede verse afectado por temperatura y envejecimiento, alterando las lecturas con el tiempo. |

### Tecnología Software: Todo lo correspondiente a desarrollo.

#### Backend

* **Node.js + Express**
* **Descripción**: Framework backend en JavaScript basado en eventos y asincronía

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Excelente integración con frontend JS (React, Vue) | Arduino UNO/Leonardo requieres módulo WiFi para comunicarse con la API |
| Alto rendimiento para proyectos IoT | No incluye sincronización en tiempo real nativo |
| Compatible con ESP32 vía HTTP o MQTT |  |

* **Python + Fast API**
* **Descripción**: Framework moderno, asincrónico y liviano para APIs en Python.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Código limpio y validación automática | Arduino UNO/Leonardo necesitan WiFi externo para enviar datos (no se conectan directamente) |
| Muy rápido en un entorno asincrónico | Comunidad más pequeña que Node.js |
| Compatible con ESP32 vía HTTP |  |

* **Firebase (realtime database)**
* **Descripción**: Backend y base de datos en la nube con sincronización en tiempo real.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Comunicación directa desde ESP32 mediante REST o MQTT | Arduino UNO/Leonardo requieren módulos WiFi-externos |
| Integración sencilla con frontend | Alta dependencia al ecosistema Google |
| Tiempo real sin configuración adicional |  |

#### Base de Datos:

* **MongoDB**
* **Descripción**: Base de datos NoSQL orientada a documentos (JSON), ideal para datos flexibles y desestructurados.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Excelente para estructuras de datos variables (como sensores) | No soporta relaciones complejas como una base de datos relacional |
| Integración nativa con Node.js y otras tecnologías modernas | Menor consistencia en operaciones críticas si no se configura correctamente. |
| Alta escalabilidad horizontal |  |

* **PostgreSQL**
* **Descripción**: Base de datos relacional de código abierto, robusta y ampliamente utilizada en entornos productivos.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Ideal para estructuras de datos bien definidos | Menor flexibilidad en esquemas cambiantes |
| Potente en consultas complejas, relaciones y validaciones | Curva de aprendizaje mayor si se busca personalización avanzada |
| Alto nivel de integridad de datos |  |

* **Firebase RTDB**
* **Descripción**: Base de datos NoSQL en la nube que permite sincronización instantánea en tiempo real.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Tiempo real nativo | Menor control sobre estructura de datos y reglas complejas |
| Sin necesidad de servidor intermedio | Altamente dependiente del ecosistema Google |
| Ideal para prototipos y apps con sincronización instantánea | Costos puedes crecer con uso intensivo |

#### FrontEnd:

* **React**
* **Descripción**: Biblioteca de JavaScript para construir interfaces web dinámicas y reactivas.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Amplia comunidad y ecosistema | Curva de aprendizaje más alta para principiantes |
| Compatible con Three.js y Babylon.js | Requiere configuración inicial más detallada |
| Ideal para proyectos modulares |  |

* **Vue.JS**
* **Descripción**: Framework progresivo para construir interfaces web reactivas

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Más fácil de aprender que React | Comunidad ligeramente más pequeña |
| Ligero y flexible | Menor ecosistema para proyectos grandes |
| Compatible con Three.js y Babylon.js |  |

#### Gráficos 3D

* **Three.js**
* **Descripción**: Librería JavaScript para crear gráficos 3D en el navegador WebGL

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Ligero y versátil | Menor nivel de abstracción, más bajo nivel que Babylon.js |
| Gran comunidad y muchos ejemplos |  |
| Integración fluida con React y Vue |  |

* **Babylon.js**
* **Descripción**: Motor gráfico 3D completo para la web.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Potente, con soporte para física, audio y animaciones complejas | Más pesado que Three.js |
| Ideal para entornos inmersivos o realistas | Requiere mayor conocimiento técnico en gráficos 3D |

#### Comunicación y servicios:

* **MQTT**
* **Descripción**: Protocolo de mensajería ligera, ideal para IoT.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Muy eficiente en dispositivos como ESP32 | Requiere un bróker MQTT (Mosquitto, HiveMQ, etc.) |
| Bajo consumo de ancho de banda | No tan directo para frontend web sin librerías adicionales |
| Ideal para entornos en tiempo real |  |

* **HTTP / REST API**
* **Descripción**: Modelo clásico donde el ESP32 envía datos como cliente a un servidor mediante solicitudes HTTP.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Fácil de implementar con ESP32, Node.js, FastAPI o Firebase | Comunicación unidireccional |
| Total, compatibilidad con backend y frontend | No es ideal para tiempo real; funciona en intervalos |

* **WebSocket**
* **Descripción**: Protocolo que permite comunicación bidireccional persistente entre cliente y servidor.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| Ideal para actualizaciones en tiempo real del estado de la planta | Requiere mayor complejidad en la gestión de conexiones |
| Compatible con React, Vue y backend modernos | No es nativo en ESP32, necesita librerías específicas |

# ANÁLISIS COMPARATIVO CUALITATIVO (IOT / HARDWARE)

## Placas programables

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **Arduino UNO** | **Arduino Leonardo** | **ESP32** |
| Documentación | Muy Alta | Alta | Alta |
| Rendimiento | Bajo | Bajo | Alto |
| Comunidad | Muy Activa | Activa | Muy Activa |
| Costo | Bajo | Mas alto | Muy Bajo |
| Escalabilidad | Limitada | Limitada | Alta |
| Compatibilidad | Altamente compatible | Compatible pero menos popular shields | Altamente compatible requiere lógica 3.3v |
| Dependencias a otras tecnologías | Baja | Baja | Media |

## Módulos / Sensores de Temperatura

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **Módulo KY-001** | **DHT11** | **DS18B20** |
| Documentación | Amplia y clara | Muy Extensa | Amplia y clara |
| Rendimiento | Preciso | Básico | Muy preciso |
| Comunidad | Activa | Muy Activa | Muy activa |
| Costo | Medio | Muy Bajo | Medio |
| Escalabilidad | Alta | Limitada | Alta |
| Compatibilidad | UNO, LEONARDO, ESP32 | UNO, LEONARDO, ESP32 | UNO, LEONARDO, ESP32 |
| Dependencias a otras tecnologías | Alta (One Wire) | Media (librería DHT) | Alta (One Wire) |

## Módulos / Sensores de Humedad

|  |  |  |
| --- | --- | --- |
| **Criterio** | **Módulo FC-28** | **Módulo Capacitivo** |
| Documentación | Muy Extensa | Buena |
| Rendimiento | Menor precisión sensible a corrosión | Más preciso, resistente |
| Comunidad | Muy Activa | Activa y creciente |
| Costo | Muy Bajo | Bajo |
| Escalabilidad | Limitada, se degrada con uso | Alta, más duradero |
| Compatibilidad | UNO, LEONARDO, ESP32 | UNO, LEONARDO, ESP32 |
| Dependencias a otras tecnologías | Baja | Baja |

## Módulos / Sensores Barométricos (Presión atmosférica)

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **BME280** | **BMP280** | **BMP180** |
| Documentación | Amplia y clara | Amplia y clara | Moderada |
| Rendimiento | Muy preciso | preciso | Aceptable |
| Comunidad | Muy activa | Activa | Moderada |
| Costo | Medio - Alto | Medio | Bajo |
| Escalabilidad | Alta | Alta | Media |
| Compatibilidad | UNO, LEONARDO, ESP32 | UNO, LEONARDO, ESP32 | UNO, LEONARDO, ESP32 |
| Dependencias a otras tecnologías | Media (I2C / SPI) | Media (I2C / SPI) | Baja (I2C / SPI) |

## Módulos / Sensores de luminosidad

|  |  |  |
| --- | --- | --- |
| **Criterio** | **BH1750** | **Fotoresistivo** |
| Documentación | Amplia y clara | Moderada |
| Rendimiento | Muy preciso | Aceptable |
| Comunidad | Activa | Activa |
| Costo | Medio | Muy bajo |
| Escalabilidad | Alta | Media |
| Compatibilidad | UNO, LEONARDO, ESP32 | UNO, LEONARDO, ESP32 |
| Dependencias a otras tecnologías | Media (I2C) | Bajo (solo ADC) |

# ANÁLISIS COMPARATIVO CUALITATIVO (SOFTWARE)

## Bases de datos

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **MongoDB** | **PostgreSQL** | **Firebase (Realtime DB)** |
| Documentación | Muy Amplia | Muy extensa | Oficial |
| Rendimiento | Alto | Muy Alto | Muy alto, tiempo real |
| Comunidad | Muy Activa | Muy Activa | Muy activa |
| Costo | Gratis (versión community) | Gratis (open source) | Gratis con límite de uso |
| Escalabilidad | Alta | Alta | Muy alta |
| Compatibilidad | Alta | Alta | Alta |
| Dependencias a otras tecnologías | Baja | Baja | Alta (dependencia con Google) |

## BackEnd

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **Node.js + Express** | **Python + Fast API** | **Firebase (Realtime DB)** |
| Documentación | Muy Amplia | Muy Buena | Muy buena |
| Rendimiento | Alto | Muy Alto | Muy alto, tiempo real |
| Comunidad | Muy Activa | Activa y creciente | Muy activa |
| Costo | Gratis (Open source) | Gratis (open source) | Gratuito con limite |
| Escalabilidad | Alta | Alta | Muy alta |
| Compatibilidad Arduino UNO | Limitada requiere usar WiFi externo | Limitada Requiere puente WiFi externo | Limitada requiere modulo WiFi |
| Compatibilidad Arduino Leonardo | Limitada requiere usar WiFi externo | Limitada Requiere puente WiFi externo | Limitada requiere modulo WiFi |
| Compatibilidad ESP32 | Directa con ESP32 vía HTTP o MQTT | Directa con ESP32 vía HTTP | Excelente: SDK y REST para ESP32 |
| Dependencias a otras tecnologías | Baja | Baja | Alta (dependencia con Google) |

## FrontEnd

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Criterio** | **React** | **Vue.js** | **Three.js** | **Babylon.js** |
| Documentación | Muy Amplia | Muy Buena | Muy buena | Muy buena |
| Rendimiento | Alto | Alto | Muy alto | Muy Alto (webGL) |
| Comunidad | Muy Activa | Muy Activa y creciente | Activa | Menos Activa |
| Costo | Gratis (Open source) | Gratis (open source) | Gratis (open source) | Gratis (open source) |
| Escalabilidad | Alta | Alta | Alta | Alta |
| Compatibilidad | Ideal con Node.js | Compatible con muchos entornos | Usable con React/Vue/JS | Compatible con JS puro y frameworks |
| Dependencias a otras tecnologías | Media | Media | Baja | Media |

### Comunicación y servicios

|  |  |  |  |
| --- | --- | --- | --- |
| Criterio | MQTT | HTTP / REST API | WebSocket |
| Documentación | Muy Buena | Muy extensa | Buena |
| Rendimiento | Muy Alto | Medio | Alto |
| Comunidad | Activa | Muy Activa | Activa |
| Costo | Gratis (Brokers) | Gratis | Gratis |
| Escalabilidad | Alta con bróker | Alta | Alta |
| Compatibilidad | Ideal para IoT | Compatible con casi todo | Compatible con frontend / backend |
| Dependencias a otras tecnologías | Requiere bróker MQTT | Baja | Requiere soporte |

### Métodos de Transmisión y visualización de datos

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Método** | **¿Requiere internet?** | **Latencia / Tiempo Real** | **Complejidad Técnica** | **Confiabilidad** | **Ventajas** | **Desventajas** | **Ideal para…** |
| WebSocket (Access Point ESP32) | No | Muy baja latencia  (<100 ms) | Media-baja | Alta | Funciona sin internet  Página web servida desde el ESP32 | Alcance limitado (10 - 20 m). Solo accesible desde la misma red.  Además, la página la sirve el ESP32 lo que conlleva a manejar el espacio de la ESP32 | Demo sin internet, prototipos locales, presentaciones. |
| MQTT + Broker | Sí | Baja latencia  (~200 ms) | Media | Muy alta | Estándar profesional de IoT, Puede funcionar en red local o nube. | Requiere configuración de broker. No es visual | Proyectos IoT distribuidos, soluciones profesionales. |
| Firebase | Sí | Baja latencia (casi tiempo real) | Baja | Media | Fácil de implementar. Integración sencilla con web | Requiere internet estable. Limitaciones de uso | Prototipos rápidos, aplicaciones conectadas a la nube |
| Servidor propio | Sí | Variable (depende del hosting) | Alta | Alta | Control total sobre datos y estructura | Mayor complejidad técnica. Configuración de backend y base de datos. | Proyectos comerciales o soluciones escalables. |
| HTTP | Sí | Alta latencia (no tiempo real) | Baja | Media-baja | Implementación muy simple. compatible con casi cualquier servidor | No ofrece tiempo real. Carga innecesaria al servidor. | Pruebas simples, monitoreo ocasional. |
| GSM | No | Media-baja latencia | Media-alta | Alta | Funciona donde no hay WiFi, permite conexión remota | Mayor consumo energético depende de la cobertura móvil. costoso | IoT remoto, monitoreo en zonas sin WiFi |

# ANÁLISIS COMPARATIVO CUANTITATIVO

Para realizar el análisis cuantitativo se utilizará principalmente el aspecto de: Precio, Energía (placas y sensores).

## PLACAS PROGRAMABLES

|  |  |
| --- | --- |
| **Placas Programables** | Precio |
| **Arduino UNO** | $14.990 |
| **Arduino Leonardo** | $39.990 |
| **ESP32** | $9.990 |

Fuente:

**Placa Arduino UNO**: <https://www.mechatronicstore.cl/arduino-uno-r3/>

**Placa Arduino Leonardo**:

<https://www.casaroyal.cl/arduino-uno-r3--mci00756/p?idsku=113330&gad_campaignid=19764436215>

**Placa ESP32**:

<https://www.falabella.com/falabella-cl/product/130037662/Esp32-Wifi-Bluetooth-Devkit-v1-30-pines-compatible-con-Arduino/130037663?kid=shopp198fc&gad_campaignid=18851715298>

## Módulos / Sensores de Temperatura

|  |  |
| --- | --- |
| **Sensores** | Precio |
| **Módulo KY-001** | $3.875 |
| **DHT11** | $2.000 |
| **DS18B20** | $2.500 |

Fuente:

**Sensor temperatura KY-001**:

<https://www.electroart.cl/2061/sensor-temperatura-ds18b20-ky-001/>

**Sensor DHT11**:

<https://ardumotica.cl/producto/sensor-digital-temperatura-y-humedad-dht11/>

**Sensor DS18B20**:

<https://afel.cl/products/sensor-temperatura-ds18b20-1-metro>

## Módulos / Sensores de humedad

|  |  |
| --- | --- |
| **Sensores** | Precio |
| **FC - 28** | $2.569 |
| **Capacitive Soil** | $1.800 |

Fuente:

**Sensor FC-28**:

<https://altronics.cl/sensor-humedad-suelo-yl-69>

**Sensor Capacitive Soil :**

<https://hubot.cl/producto/sensor-humedad-del-suelo-capacitivo-33-5v-sku-5004/>

## Módulos / Sensores Barométricos

|  |  |
| --- | --- |
| **Sensores** | Precio |
| **BME 280** | $7.900 |
| **BMP 280** | $2.000 |
| **BMP 180** | $2.300 |

Fuente:

**Sensor BME 280**:

<https://hubot.cl/producto/sensor-de-presion-temperatura-y-humedad-bme280-sku-653h2/>

**Sensor BMP 280**:

<https://afel.cl/products/sensor-barometrico-bmp280?variant=45125221384344&country=CL&currency=CLP>

**Sensor BMP 180**:

<https://afel.cl/products/sensor-barometrico-bmp180>

## Módulos / Luminosidad

|  |  |
| --- | --- |
| **Sensores** | Precio |
| **BH1750** | $1.856 |
| **Fotoresistivo** | $2.490 |

Fuente:

**Sensor BH1750**:

<https://es.aliexpress.com/item/1005008661005379.html?spm=a2g0o.productlist.main.1.1a243a33ONrDE4&algo_pvid=f39e0c92-1bf4-4b95-bf7b-66987d003208&pdp_ext_f=%7B%22order%22%3A%2233%22%2C%22eval%22%3A%221%22%2C%22fromPage%22%3A%22search%22%7D&utparam-url=scene%3Asearch%7Cquery_from%3A%7Cx_object_id%3A1005008661005379%7C_p_origin_prod%3A>

**Sensor Fotoresistivo**:

<https://www.mercadolibre.cl/sensor-luminosidad-arduino-fotoresistivofotorresistencia/up/MLCU92436908?pdp_filters=item_id:MLC472107069>

## Métrica General Estándar (MGE) para comparación de Bases de Datos

El propósito es evaluar y contrastar el costo mensual de una configuración de bajo consumo o destinada a un proyecto pequeño en MongoDB y PostgreSQL, junto con el costo inicial de uso en Firebase.

(Implementación en la nube)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Base de Datos (Tecnología)** | **Nivel de Servicio** | **Precio Mensual Estimado (USD)** | **Precio Mensual Estimado**  **(CLP)** | **Asignación de Recursos (Capacidad)** | **Modelo de Costo Clave** | **Ratio Precio/ GB RAM (USD/mes)** |
| **Mongodb**  **(Dedicated)** | M10 (Entrada) | $60.80 | $58.510 | 2 vCPUs /2 GB RAM / 10 GB Storage | Fijo por Recurso | $30.40 |
| **PostgreSQL**  **(DigitalOcean)** | Managed (Básico) | $15.15 | $14.582 | 1 vCPU /1 GB RAM / 10 GB Storage | Fijo por Recurso | $15.15 |
| **Firebase (Realtime DB)** | Plan Blaze (Pago por uso) | $10.00 a $30.00 | $9.624 a $28.872 | Variable (Operaciones, GB Transferidos) | Variable por Uso | N/A |

## Métodos de Transmisión y visualización de datos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Método | **Costo inicial** | **Detalle gasto inicial** | **Costo mensual** | **Detalle gasto mensual** |
| WebSocket Local | 9.500 - 28.500 CLP | ESP32 y sensores | 0 | Ninguno |
| MQTT + Broker | 9.500 - 28.500 CLP | ESP32 y sensores | 0 - 19.000 CLP | Plan de broker en la nube |
| Firebase | 9.500 - 28.500 CLP | ESP32 y sensores | 0 - 47.500 CLP | Plan de Firebase si supera límites gratuitos |
| Servidor propio | 9.500 - 28.500 CLP + VPS / dominio | ESP32 y sensores; VPS y dominio | 14.250 - 28.500 CLP | Hosting VPS, base de datos |
| HTTP Polling | 9.500 - 28.500 CLP | ESP32 y sensores | 0 - 4.750 CLP | Hosting básico opcional |
| GSM/4G | 19.000 - 47.500 CLP | ESP32 + módulo GSM/4G | 4.750 - 19.000 CLP | Plan de datos móviles |

(Implementación en servidor físico)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Proveedor / Modelo** | **Costo Inicial (CLP)** | **Detalle gasto inicial** | **Costo Mensual (CLP)** | **Detalle gasto mensual** | **CPU/RAM/Almacenamiento** |
| **Intel NUC / Mini PC** | 250.000 - 400.000 | Equipo compacto con SSD, RAM y gabinete | 2.000 - 5.000 | Electricidad ligera | 4 núcleos físicos / 8 GB RAM / 256 GB SSD |
| **HP ProDesk / Dell OptiPlex** | 350.000 -  500.000 | Equipo completo, almacenamiento interno | 3.000 - 6.000 | Electricidad | 4-6 núcleos físicos / 8-16 GB RAM / 256 - 512 GB SSD |
| **Servidor rack / HPE ProLiant MicroServer** | 500.000 -  900.000 | Rack, HDD/SSD, RAM, fuente de poder | 5.000 -  10.000 | Electricidad, refrigeración ligera | 8 núcleos físicos / 16 GB RAM / 1 TB HDD |
| **Servidor dedicado / HPE ProLiant ML350** | 1.500.000 -  2.500.000 | Rack, Almacenamiento, RAM, gabinete | 10.000 - 20.000 | Electricidad, refrigeración,  mantenimiento | 16 núcleos físicos / 32 GB RAM / 2 TB SSD |

## Servidor virtual (VPS)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Proveedor / Plan** | **Costo Inicial (CLP)** | **Detalle gasto inicial** | **Costo Mensual (CLP)** | **Detalle gasto mensual** | **CPU/RAM/Almacenamiento** |
| **SoluciónHost VPS 1** | 2.500 | Configuración inicial mínima | 2.500 | Mantención básica, electricidad | 1 vCPU / 1 GB RAM / 20 GB SSD |
| **PremiumHosting VPS Básico** | 9.900 | Configuración VPS, instalación base | 9.900 | Electricidad, hosting, mantenimiento | 2 vCPU / 4 GB RAM / 25 GB SSD |
| **Xhost Starter** | 58.250 | Configuración VPS y instalación | 58.250 | Electricidad, hosting, mantenimiento | 6 vCPU / 6 GB RAM /500 GB SSD |

## Servidor Cloud

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Proveedor / Plan** | **Costo Inicial (CLP)** | **Detalle gasto inicial** | **Costo Mensual (CLP)** | **Detalle gasto mensual** | **CPU / RAM / Almacenamiento** |
| **AWS EC2 t3a.medium** | 0 | Ninguno | 30.000 - 50.000 | uso de VM, almacenamiento, tráfico | 2 vCPU / 4GB RAM / 50 GB SSD |
| **Google Cloud VM e2-medium** | 0 | Ninguno | 28.000 - 45.000 | Uso de VM, almacenamiento, tráfico | 2 vCPU / 4 GB RAM / 50 GB SSD |
| **Azure B2ms** | 0 | Ninguno | 30.000 - 55.000 | Uso de VM, almacenamiento, tráfico | 2 vCPU / 4 GB RAM / 80 GB SSD |

## PLACAS CUANTITATIVO

### Consumo de Energía (mA)

|  |  |  |
| --- | --- | --- |
| **Placa** | **Activo** | **Deep Sleep** |
| **Arduino UNO** | 50 | 0.5 |
| **Arduino Leonardo** | 45 | 0.5 |
| **ESP32-S3** | 80 - 120 | 10 - 50 |

### Rendimiento / CPU

|  |  |
| --- | --- |
| Placa | Frecuencia CPU (Mhz) |
| Arduino UNO | 16 |
| Arduino Leonardo | 16 |
| ESP32-S3 | 240 |

### Almacenamiento / Memoria

|  |  |  |  |
| --- | --- | --- | --- |
| Placa | RAM | Flash | EEPROM |
| Arduino UNO | 2 KB | 32 KB | 1 KB |
| Arduino Leonardo | 2.5 KB | 32 KB | 1 KB |
| ESP32-S3 | 512 KB SRAM + 384 KB RTC | 4 - 8 MB | Emulado en Flash |

## SENSORES CUANTITATIVO (SENSOR DE TEMPERATURA)

### Consumo de Energía (mA)

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Consumo Activo** | **Deep Sleep** |
| **KY - 001** | 1.0 | <1.0 |
| **DHT11** | 0.3 | 0.06 |
| **DS18B20** | 1.5 | ~0.001 |

### Rendimiento (CPU)

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Resolución** | **Tiempo de Respuesta** |
| **KY - 001** | 9 bits | 750 ms |
| **DHT11** | 8 bits | 1-2 s |
| **DS18B20** | 9 - 12 bits | 750 ms-1 s |

## SENSORES CUANTITATIVO (SENSOR DE HUMEDAD)

### Consumo de Energía (mA)

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Consumo Activo** | **Deep Sleep** |
| **FC-28** | 10 | ~ 0 |
| **Capacitive Soil** | 0.15 - 0.3 | <0.01 |

### Rendimiento (CPU)

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Resolución** | **Tiempo de Respuesta** |
| **FC-28** | Análogico / NA | 10 - 100 ms |
| **Capacitive Soil** | Análogico / NA | 50 - 500 ms |

## SENSORES CUANTITATIVO (SENSOR DE BAROMETRICIDAD)

### Consumo de Energía (mA)

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Consumo Activo** | **Deep Sleep** |
| **BME280** | 0.02 | 0.00002 |
| **BMP280** | 0.0027 | 0.0000027 |
| **BMP180** | 0.005 | 0.0000005 |

### Rendimiento (CPU)

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Resolución** | **Tiempo de Respuesta** |
| **BME280** | 16 bits | 1.25 ms |
| **BPM280** | 16 bits | 1.25 ms |
| **BPM180** | 16 bits | 5 - 10 ms |

## SENSORES CUANTITATIVO (SENSOR DE LUMINOSIDAD)

### Consumo de Energía (mA)

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Consumo Activo** | **Deep Sleep** |
| **BH1750** | 0.12 - 0.18 | 0.0005 (~0) |
| **Fotoresistivo** | 0.01 - 0.05 | ~0 |

### Rendimiento (CPU)

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Resolución** | **Tiempo de Respuesta** |
| **BH1750** | 16 bits | 120 - 180 ms |
| **Fotoresistivo** | Depende del ADC | < 10 ms |

# Selección de tecnologías (IOT / Hardware)

Para seleccionar las tecnologías IOT se contemplarán 3 parámetros importantes: Factibilidad Técnica, Factibilidad Económica, Factibilidad Implementativa.

## Factibilidad Técnica: Se analizan precisión, compatibilidad y facilidad de integración de los sensores y la placa seleccionada

|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Precisión / Rendimiento** | **Compatibilidad** | **Facilidad integración** |
| **ESP32-S3** | CPU 240 MHz, WiFi + Bluetooth integrados, alta velocidad y memoria | Sensores digitales y analógicos | Programación mediante arduino IDE o PlatformIO, soporte amplio. |
| **DS18B20** | Muy precisa (9 - 12 bits, 750 ms - 1 s) | Arduino UNO, Leonardo, ESP32 | Protocolo 1-Wire, requiere librería específica |
| **BMP280** | Preciso (16 bits, 1.25 ms) | Arduino UNO, Leonardo, ESP32 | I2C / SPI, fácil de integrar |
| **BH1750** | Muy preciso (16 bits, 120 - 180 ms) | Arduino UNO, Leonardo, ESP32 | I2C, sencillo de usar |
| **Capacitive Soil** | Más preciso que FC-28, estable | Arduino UNO, Leonardo, ESP32 | Analógico, lectura directa por ADC |

### Conclusión factibilidad técnica:

La ESP32-S3 resulta totalmente compatible con todos los sensores seleccionados, ofreciendo además conectividad inalámbrica integrada. Los sensores presentan un rendimiento adecuado y facilitan la integración. Solo el DS18B20 requiere un manejo particular del protocolo 1-wire, pero puede ser gestionado mediante librerías disponibles.

## Factibilidad Económica:

Se consideran el precio de los sensores y su consumo energético, incluyendo la placa ESP32:

|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Precio CLP** | **Consumo Activo (mA)** | **Deep Sleep (mA)** |
| **ESP32** | $ 9.990 | 80 - 120 | 10 - 50 |
| **DS18B20** | $2.500 | 0.05 | <0.0005 |
| **BMP280** | $2.000 | 0.0027 | 0.0000027 |
| **BH1750** | $1.856 | 0.12 - 0.18 | 0.0005 (~0) |
| **Capacitive Soil** | $1.800 | 0.15 - 0.3 | <0.01 |

### Conclusión factibilidad económica:

El costo total de los sensores junto con la placa es accesible (~18.146 CLP). Además, el consumo energético de todos los componentes es bajo, especialmente en modo Deep sleep.

## Factibilidad Implementativa

Se evalúa la disponibilidad de librerías, soporte comunitario y facilidad de programación:

|  |  |  |
| --- | --- | --- |
| **Componente** | **Librerías / Comunidad** | **Nivel de complejidad** |
| **ESP32** | Muy amplia, soporte en Arduino IDE, PlatformIO y MicroPython | Bajo - medio (WiFi / Bluetooth + ADC / I2C / 1-wire) |
| **DS18B20** | Muy buena, Arduino y ESP32 | Medio (1 - Wire) |
| **BMP280** | Amplia documentación y muy utilizado | Bajo |
| **BH1750** | Amplia documentación y muy utilizado | Bajo |
| **Capacitive Soil** | Muy usado, ejemplos disponibles. | Bajo |

### Conclusión implementativa:

Todos los sensores cuentan con soporte suficiente para su uso con la ESP32. La curva de aprendizaje es baja o media, siendo el DS18B20 el único que requiere mayor atención al protocolo 1-Wire, sin que esto represente una limitación significativa.

## Selección de tecnologías (Software)

Para seleccionar las tecnologías de software se contemplarán 3 parámetros importantes: Factibilidad Técnica, Factibilidad Operativa, Factibilidad Implementativa.

### Factibilidad Técnica

Se analiza compatibilidad, precisión, rendimiento y facilidad de integración entre hardware y software:

|  |  |  |
| --- | --- | --- |
| **Componente** | **Compatibilidad / Rendimiento** | **Facilidad de integración** |
| **Node js + Express** | Amplia compatibilidad con ESP32 via HTTP o WebSocket | Fácil de integrar con frontend React y bases de datos como InfluxDB |
| **Influxdb** | Optimizada para datos de sensores, series temporales | Compatible con Node js, consulta rápida de datos histórico y en tiempo real |
| **WebSocket** | Comunicación bidireccional y en tiempo real | Compatible con Node js y React requiere librerías específicas. |
| **React + Three.js** | Visualización web dinámica y modelos 3D interactivos | Amplia documentación, integración fluida con Node js y WebSocket |

#### Conclusión técnica:

Todas las tecnologías seleccionadas son compatibles entre sí y con el hardware (ESP32 + sensores). Permiten la transmisión, almacenamiento y visualización de datos de forma eficiente, incluyendo el modelo 3D en tiempo real.

### Factibilidad Operativa

Se analiza qué tan adecuado es el software para cumplir los objetivos del proyecto y las necesidades de monitoreo de la planta:

|  |  |  |
| --- | --- | --- |
| **Componente** | **Operatividad / Escenarios** | **Ventajas** |
| **Node js + Express** | Servidor backend robusto para IoT | Permite manejar múltiples sensores y solicitudes simultáneas |
| **InfluxDB** | Base de datos para series temporales | Consulta rápida de históricos, ideal para gráficos y análisis |
| **WebSocket** | Comunicación en tiempo real | Actualización instantánea de los valores de los sensores en el frontend |
| **React +** [**Three.js**](http://three.js) | Interfaz de usuario y visualización 3D | Permite representar la planta y el estado de sensores de manera interactiva |

#### Conclusión operativa:

La combinación de Node js + Express, InfluxDB, WebSocket y React + Three js permite un sistema de monitoreo en tiempo real eficiente, escalable y con visualización interactiva, cumpliendo los objetivos del proyecto.

## Factibilidad Implementativa

Se analiza facilidad de programación, disponibilidad de librerías y soporte:

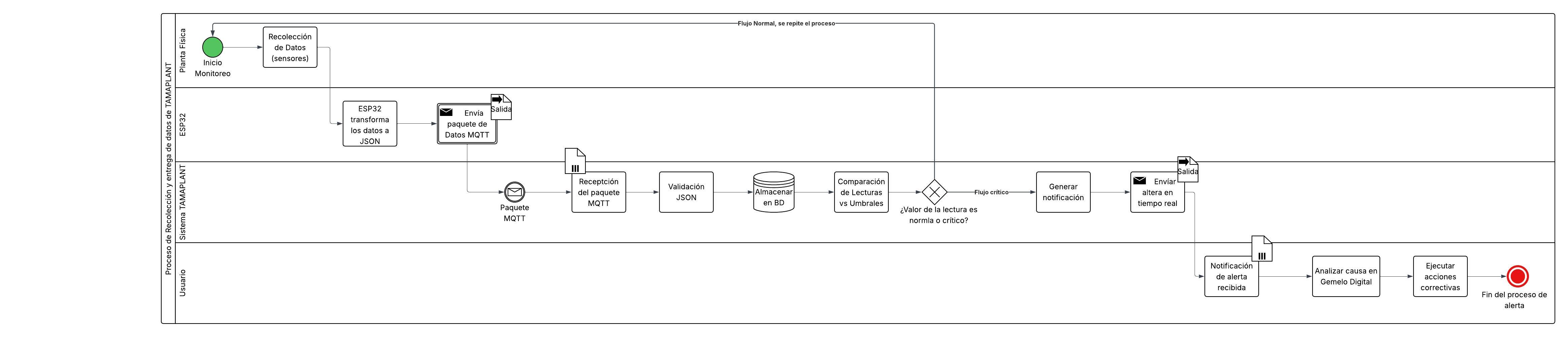
|  |  |  |
| --- | --- | --- |
| **Componente** | **Librerías / Comunidad** | **Nivel de complejidad** |
| **Node js + Express** | Muy amplia, documentación extensa | Bajo |
| **InfluxDB** | Comunidad activa, documentación oficial | Medio (series temporales, consultas específicas) |
| **WebSocket** | Amplia documentación, ejemplos en Node js y React | Medio (requiere gestión de conexiones) |
| **React + Three js** | Comunidad amplia, tutoriales y ejemplos. | Medio (Three.js requiere conocimiento 3D) |

#### Conclusión implementativa:

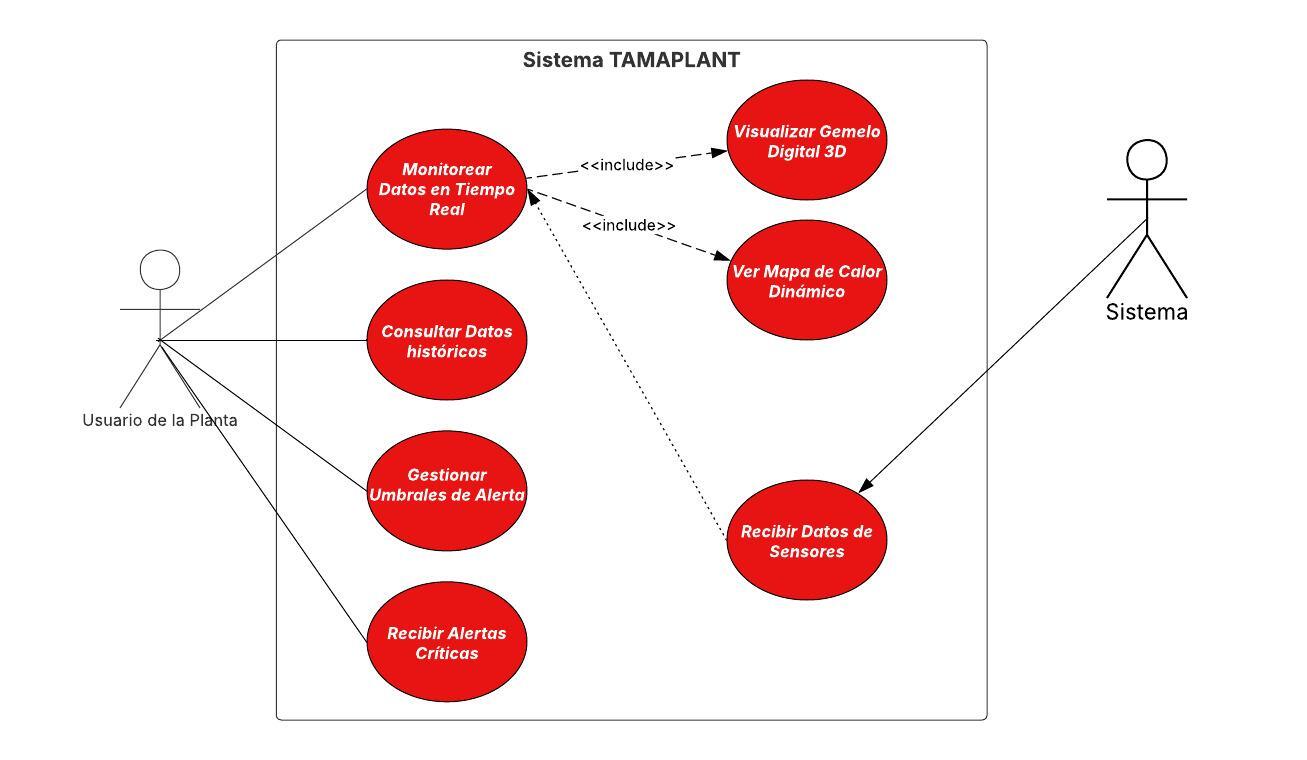
Todas las tecnologías cuentan con documentación y ejemplos suficientes para facilitar el desarrollo. La curva de aprendizaje es baja a media, siendo Three.js y la gestión de WebSocket los únicos elementos que requieren mayor atención técnica.

# DETALLE DE ARQUITECTURA A IMPLEMENTAR

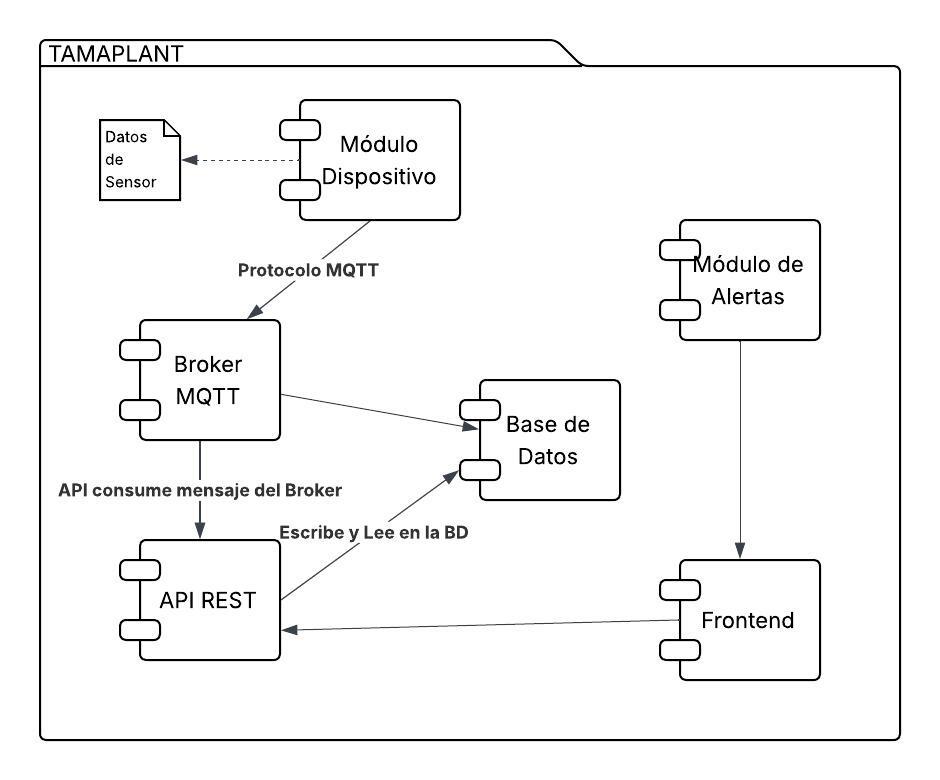
## Diagrama BPMN



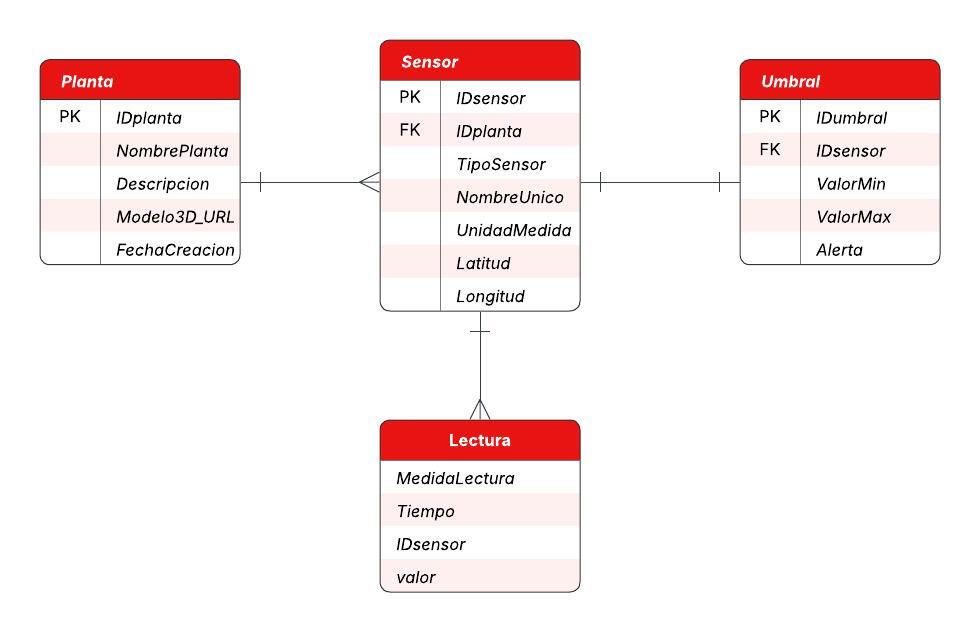
## Diagrama de Casos de Uso



## Diagrama de Componentes



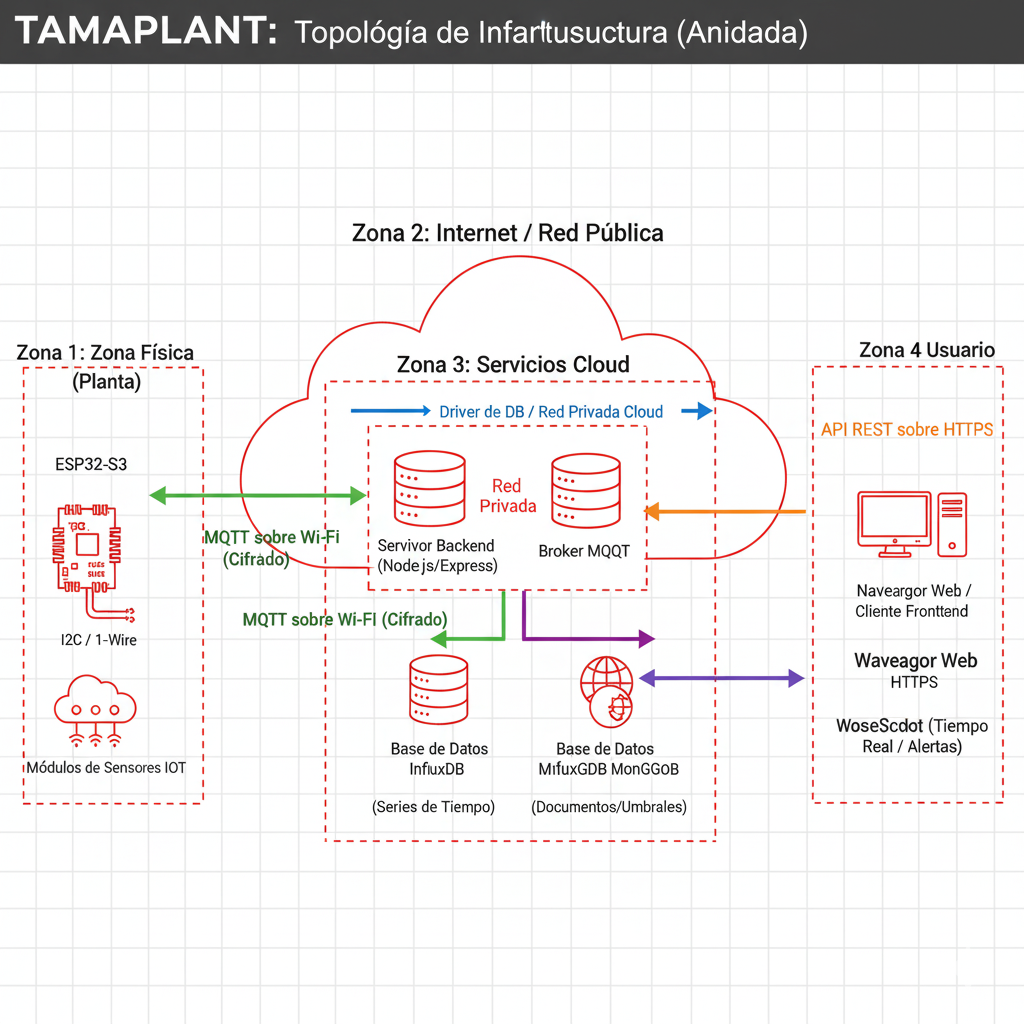
## Modelo de Datos



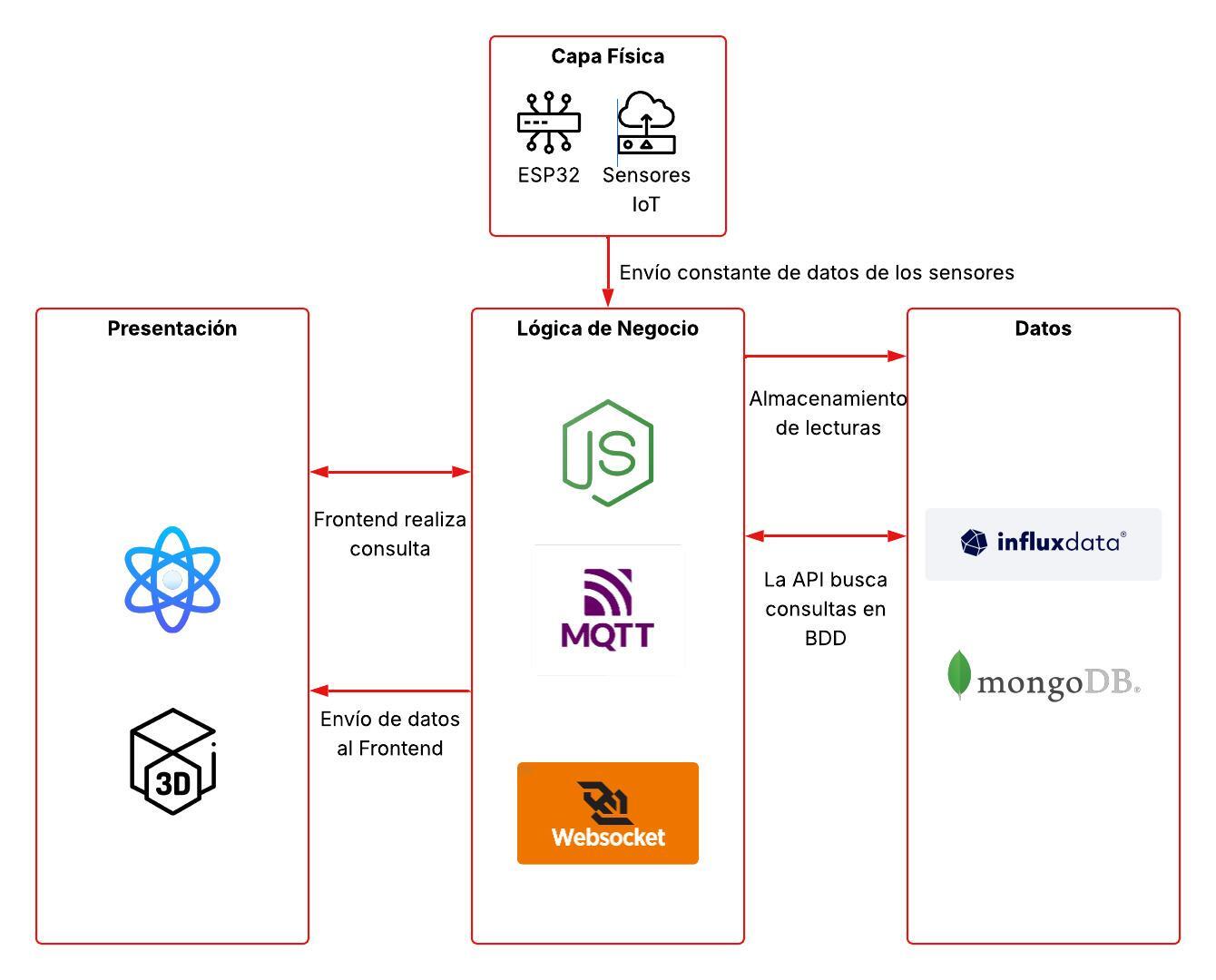
## Topología de Comunicación

## 

## Topología de Infraestructura



## Topología de Arquitectura



# IMPLEMENTACIÓN DE LOS KPI Y SLA

## Descripción de los KPI

|  |  |  |  |
| --- | --- | --- | --- |
| **KPI** | **Métrica** | **Valores Objetivo** | **Impacto al Proyecto** |
| **MTTD** | Tiempo desde que el sensor registra un valor crítico hasta que el sistema envíe la alerta | MTTD menor a 5 segundos | El objetivo es reducir la reacción de minutos a segundos |
| **UPTIME** | Porcentaje de tiempo que la plataforma y el pipeline de datos están operativos | Disponibilidad de un 99,5% | Mide la robustez de la arquitectura Cloud, y es una medida crítica para el monitoreo constante |
| **Latencia de Visualización de Datos** | Tiempo entre la lectura del sensor y la visualización en el Gemelo Digital | Latencia menor o igual a 3 segundos | Mide la calidad de la experiencia del usuario con el frontend, asegurando que el modelo 3D refleje el estado real de la planta |
| **Tasa de Errores de Conexión IoT** | Porcentaje de paquetes de datos perdidos o inválidos respecto al total enviado | Tasa de error menor al 0,1% | Mide la estabilidad de la red y el firmware del ESP32. Un error bajo asegura la integridad de la información |

## Descripción de SLA

|  |  |  |
| --- | --- | --- |
| **SLA** | **Descripción del Nivel de Servicio** | **Consecuencia del Incumplimiento** |
| **Tiempo de Respuesta del Gemelo Digital** | El frontend debe cargar y renderizar completamente el modelo 3D y los datos iniciales en un máximo de 5 segundos. | Impacto negativo en la usabilidad y percepción del rendimiento. |
| **Acceso a Datos Históricos** | Se garantiza la consulta y descarga de datos de sensores almacenados por un periodo establecido. | Pérdida de la capacidad de realizar análisis de tendencia a largo plazo, limitando el valor investigado. |
| **Capacidad de Alerta** | El módulo de alertas debe notificar al usuario en un tiempo menor a 5 segundos cuando se cruza un umbral preestablecido. | Retraso en la toma de decisiones, lo que puede resultar en daños en la planta y/o pérdida de recursos. |
| **Ventana de Mantenimiento** | Se realizará una ventana de mantenimiento fuera del horario de uso general con una duración máxima de 2 horas una vez al mes. | Interrupción de la recolección de datos y riesgo de incumplimiento del KPI de Disponibilidad. |

# PLAN DE PRUEBAS Y ASEGURAMIENTO DE CALIDAD

## Plan de Pruebas

El plan se divide por componentes, asegurando que cada capa funcione de manera correcta de forma aislada y en conjunto.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de Prueba** | **Componentes** | **Caso de Prueba** | **Resultado Esperado** |
| **Pruebas Unitarias** | ESP32-S3 y Sensores | Simular una temperatura fuera de rango normal en el sensor y verificar que ESP32 lo lee correctamente y lo formatea a JSON | El ESP32 publica un JSON válido con la temperatura. |
| **Prueba de Integración** | ESP32 → bróker MQTT → Node.js →BD | Enviar 100 paquetes de datos por MQTT y verificar que los 100 registros se escriben correctamente en la base de datos | La base de datos contiene los 100 nuevos registros y el KPI de Tasa de Errores menor al 0,1%. |
| **Pruebas del Sistema** | Lógica de Umbrales y Módulo de alertas | Configurar un umbral de humedad menor al 30%. Simular un valor de 29% y verificar las acciones tomadas por el sistema | El Módulo de Alertas dispara un evento WebSocket al Frontend y el Navegador Web muestra una notificación. |
| **Pruebas de Usabilidad** | React / Three.js | Un usuario interactúa con el Modelo 3D y hace clic en el sensor | El modelo 3D gira fluidamente y al hacer clic, aparece un popup con los datos en tiempo real. |

## Normas y Estándares

Se adoptarán estándares para garantizar la calidad, el mantenimiento y la seguridad del código.

1. Estándar de Codificación: Se utilizará ESLint para forzar un estilo de código consistente en JavaScript, mejorando la legibilidad y facilitando el code review.
2. Protocolo de Mensajería: Se implementará el estándar MQTT v3.1.1 para comunicación IoT. Se utilizará QoS Nivel 1 para asegurar que la mayoría de los mensajes lleguen, minimizando la pérdida de datos.
3. Seguridad y Conectividad: La comunicación entre el ESP32 e Internet se realizará mediante el protocolo seguro TLS/SLL sobre MQTT, cifrando el tráfico de datos.
4. Modelado de Datos: Se seguirán las directrices de la Notación Pata de Gallo para el Modelo de Datos y el esquema de Tag-Field-Timestamp para las series de tiempo.

# PLAN DE IMPLEMENTACIÓN

## Gestión de Disponibilidad

|  |  |  |
| --- | --- | --- |
| **Área** | **Actividad de Disponibilidad** | **Herramienta / Estratégia** |
| **Infraestructura Cloud** | Monitoreo del rendimiento y salud del Servidor Backend | Prometheus/Grafana o servicio de monitoreo Cloud para vigilar el uso de CPU, memoria y ancho de banda. |
| **Base de Datos** | Asegurar la conectividad constante y el espacio de almacenamiento de InfluxDB. | Configuración de réplicas de DB y alertas de disco lleno. |
| **IoT Edge** | Detección de desconexiones de microcontroladores. | Uso de la característica “Last Will and Testament” de MQTT para alertar inmediatamente si el ESP32 pierde la conexión. |

## Gestión de Continuidad

Se enfoca en la capacidad del sistema de recuperarse de fallos.

### Política de Respaldo

Implementar un respaldo incremental de los datos obtenidos de series de tiempo.

Crear un respaldo completo semanal de configuraciones, usuarios y umbrales definidos.

### Restauración del Servicio

En caso de caída total del servidor, se define un Recovery Time Objective (RTO) de 4 horas para reinstalar la aplicación Node.js y restaurar las bases de datos en una nueva instancia.

### Código Fuente

El código de firmware y backend/frontend se mantendrá en un repositorio Git remoto y seguro.

## Plan de Mantención

### Mantenimiento Preventivo de Software

Se impondrán actualizaciones trimestrales de dependencias de Node.js para incorporar parches de seguridad y rendimiento. Así como, tareas de limpieza en InfluxDB para eliminar datos obsoletos y mantener la velocidad de las consultas.

### Mantenimiento Correctivo

Revisión periódica y manual para la calibración de los sensores para evitar la deriva y garantizar la precisión de los datos. Así también, realizar chequeos semestrales del estado de la fuente de alimentación del ESP32.

### Mantenimiento Evolutivo

Documentación y planificación de nuevas funcionalidades para la fase 2 del proyecto.

# CONCLUSIONES

El proyecto TAMAPLANT no solo cumple su objetivo principal de transformar el monitoreo manual de una planta en un proceso digital, inmersivo y en tiempo real, sino que establece una base tecnológica robusta para el futuro de la Agricultura de Precisión en contextos controlados.

# REFERENCIAS BIBLIOGRÁFICAS

1. ¿Qué es un Gemelo Digital?, IBM, 2021, <https://www.ibm.com/es-es/think/topics/digital-twin>.
2. Gemelos digitales: La revolución de la agricultura de precisión, Violeta Cubero, Plataforma Tierra, 2025, <https://www.plataformatierra.es/innovacion/los-gemelos-digitales-en-la-agricultura>.
3. Internet de las cosas: la nueva revolución verde, Yolanda González Menéndez, 2018, <https://www.bbva.com/es/internet-cosas-nueva-revolucion-verde/>.
4. ‘Smart farming’: Así es la agricultura sostenible que promete transformar el campo, Communications, 2025, <https://www.bbva.com/es/sostenibilidad/smart-farming-asi-es-la-agricultura-sostenible-que-promete-transformar-el-campo/>.
5. Condiciones de trabajo en invernaderos V Región, Celina Carrasco y Patricia Vega, <https://www.dt.gob.cl/portal/1629/articles-93453_recurso_1.pdf>.
6. IoT en la agricultura: 10 casos de uso de tecnologías agrícolas inteligentes, Jayna Locke, 2025, <https://es.digi.com/blog/post/iot-in-agriculture>.
7. Cultivos del futuro: oportunidades y desafíos del Internet de las Cosas en la agricultura chilena, Universidad de O’Higgins, 2024, <https://www.uoh.cl/cultivos-del-futuro-oportunidades-y-desafios-del-internet-de-las-cosas-en-la-agricultura-chilena/>.
8. Invernaderos Inteligentes: Transformación de la agricultura mediante sensores inteligentes inalámbricos, Logicbus, 2023, <https://www.logicbus.com.mx/blog/invernaderos-inteligentes-transformacion-de-la-agricultura-mediante-sensores-inteligentes-inalambricos/>.
9. El problema de la complejidad y la integración en entornos de IoT, SEDICI, 2023, <https://sedici.unlp.edu.ar/bitstream/handle/10915/110802/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y>.