Universidad de Santiago de Chile

Facultad de Ingeniería

Departamento de Ingeniería Informática

Sistemas Distribuidos

Monitoreo Ambiental en Aulas Académicas

Entrega 3: Minimum Viable Product (MVP)

Integrantes:

- Alexandra Navarro
- Javiera Vergara
- Ariel Núñez

Profesor:

- Manuel Manriquez

Ayudante:

- Hernán Aravena

Contenido

1. Introducción

1.1. Motivación

Los entornos físicos de aprendizaje, tanto en colegios como en universidades, constituyen espacios fundamentales donde estudiantes y docentes transcurren la mayor parte de su jornada diaria. La calidad de estos espacios se convierte, por tanto, en un instrumento muy valioso para garantizar un proceso educativo efectivo. En este contexto, Chile según datos de la OCDE, es el país con el número más alto de horas cronológicas semanales dedicadas a clases, y sus docentes imparten en promedio 28.5 horas semanales de docencia directa, siendo el valor más alto entre los 49 países que participaron en TALIS 2018[1].

Esto significa que las salas de clases se transforman en el principal escenario del proceso formativo. Pero, ¿qué pasa si ese entorno no es adecuado?. Factores como la mala calidad del aire, temperaturas muy altas o muy bajas, niveles de humedad desbalanceados, ruido constante o iluminación inadecuada pueden no solo afectar la concentración, sino también la salud y el bienestar emocional de los que se encuentran allí. Un estudio elaborado el 2024 por la Universidad de Salford, en Inglaterra, reveló que unas óptimas condiciones en el aula pueden mejorar hasta un 25% el rendimiento escolar. Aspectos como la iluminación, el mobiliario, el color de las paredes, una buena climatización o incluso la instalación eléctrica e inalámbrica pueden influir en el proceso de enseñanza-aprendizaje[2]. La situación descrita es algo que no ha pasado desapercibido, ya que a través de la radio de la Universidad de Concepción se ha informado que el Centro de Desarrollo Urbano Sustentable (CEDEUS) se encuentra trabajando en el proyecto “Ambientes Educativos Resilientes y Saludables”, donde se abordan los desafíos ambientales que afectan la calidad y el rendimiento académico en los

[1] Organisation for Economic Co-operation and Development. (n.d.). *Chile – teachers and teaching conditions (TALIS 2018). Education GPS.* Recuperado el 2 de mayo de 2025, de https://gpseducation.oecd.org/CountryProfile?primaryCountry=CHL&topic=TA&treshold=5

[2] EDUCACIÓN 3.0. (2025, junio 4). *¿El diseño del aula afecta en el rendimiento académico de los alumnos?* EDUCACIÓN 3.0. https://www.educaciontrespuntocero.com/noticias/diseno-aula-rendimiento-academico-alumnos/

[3] Centro de Desarrollo Urbano Sustentable (CEDEUS). (2024, septiembre 6). *Temperatura, humedad y ventilación: Proyecto aborda los factores ambientales que inciden en el rendimiento académico en las escuelas*. CEDEUS. https://www.cedeus.cl/blog/2024/09/06/temperatura-humedad-y-ventilacion-proyecto-aborda-los-factores-ambientales-que-inciden-en-el-rendimiento-academico-en-las-escuelas/

[4] Álvarez, A. (2023, 2 de marzo). *Vuelta a clases: cómo incide la calidad del aire y la temperatura del aula en el rendimiento*. ARQA Empresas. https://arqa.com/empresas/novedades/calidad-del-aire-y-la-temperatura-del-aula.html

[5] Porras, E. (2025, abril 2). *Optimización del confort térmico y la calidad del aire interior en ambientes educativos*. ACR Latinoamérica. https://www.acrlatinoamerica.com/mas-a-fondo/otros-enfoques/20675-optimizacion-del-confort-termico-y-la-calidad-del-aire-interior-en-ambientes-educativos.html

[6] Bionoticias. (2023, junio 19). *¿Sabías que 1 de cada 51 niños es diagnosticado con autismo en Chile?* https://bionoticias.cl/sabias-que-1-de-cada-51-ninos-es-diagnosticado-con-autismo-en-chile/

colegios, enfocados en factores como la temperatura, la humedad, la calidad del aire y la ventilación[3]. Y también existe otro estudio relacionado a la temperatura y calidad del aire, donde se explica que un aula con una ventilación deficiente y altos niveles de CO2 puede reducir la productividad de los estudiantes en hasta un 20%, y además reducir la temperatura de 25ºC a 20ºC puede incrementar hasta en un 12% los resultados en pruebas de matemáticas y lenguaje, lo que refuerza la necesidad de mantener un ambiente ideal para el desarrollo de la enseñanza[4].

Un ambiente incómodo puede generar fatiga, somnolencia, dolor de cabeza e incluso, en casos más sensibles, hasta crisis sensoriales en estudiantes con condiciones específicas como el Trastorno del Espectro Autista (TEA)[5]. Según un estudio realizado por la Revista Chilena de Pediatría, actualmente, 1 de cada 51 niños es diagnosticado con este trastorno. Chile supera la prevalencia de diagnósticos en comparación a países a nivel mundial cómo Estados Unidos que reporta 1 en 59, Inglaterra con 1 en 57, Colombia con 1 en 68, España con 1 en 100 y México con 1 en 115[6]. Esta situación es incluso más crítica, ya que se toman más en consideración la cantidad de ruido y una buena construcción de la infraestructura de la sala de clases, pero más detenidamente la necesidad de que estos factores sean adecuados. Y no solo los estudiantes se ven afectados, ya que los docentes también enfrentan estas consecuencias al trabajar durante tantas horas en ambientes que no siempre se encuentran en condiciones óptimas.

Frente a esta realidad, surge la necesidad de buscar soluciones más inteligentes y adaptables. Hoy, gracias a la tecnología es posible diseñar sistemas que monitorean en tiempo real las condiciones dentro de las aulas. Un sistema distribuido de monitoreo ambiental representa una oportunidad concreta para lograr que las salas de clases sean espacios más cómodos, saludables e inclusivos, enfocado con lo que realmente necesitan los estudiantes y profesores para enseñar y obtener un buen aprendizaje. Un sistema de este estilo beneficiaría a los estudiantes principalmente, ya que con nuestra solución se podrá verificar que las salas se encuentran en las mejores condiciones de desempeño académico, para los profesores, ya que el manejo del estudiantado será más ameno debido a que los mismos estudiantes no encuentran situaciones perjudiciales sensorialmente hablando, y para los educadores de niños en el espectro autista, porque la solución permitirá verificar que los factores que medirá la aplicación se encuentren en niveles adecuados para el desarrollo de su profesión.

1.2. Introducción

Mejorar las condiciones físicas dentro de las salas de clases ya no puede considerarse un aspecto secundario, sino una necesidad. Está comprobado que factores como la temperatura, el nivel de dióxido de carbono, la humedad, el ruido y la iluminación impactan directamente en el bienestar y la concentración tanto de estudiantes como de docentes. Por ello, surge la oportunidad de aplicar soluciones tecnológicas que permitan monitorear y controlar estas variables de forma continua, facilitando una respuesta oportuna cuando las condiciones no son adecuadas.

En este contexto, se propone el desarrollo de un sistema distribuido de monitoreo ambiental, basado en una arquitectura Edge-Fog-Cloud, que permite captar, procesar y visualizar en tiempo real las condiciones ambientales dentro de las salas de clases. Este sistema se compone principalmente de sensores físicos que están conectados a microcontroladores ESP32, nodos fog instalados localmente para el proceso preliminar, y una capa en la nube para almacenamiento histórico y análisis remoto. Esta solución busca apoyar la toma de decisiones preventivas por parte de docentes y personal administrativo, con el fin de mejorar las condiciones del entorno educativo y fomentar espacios más saludables e inclusivos.

El presente informe detalla el diseño técnico del sistema propuesto, sus componentes, funcionalidades, herramientas utilizadas, políticas de operación y proyección de costos, con el objetivo de que pueda ser implementado en contextos reales, específicamente en establecimientos educacionales.

2. Objetivos

2.1. Objetivo general

Desarrollar un sistema distribuido de monitoreo ambiental en aulas académicas que permita medir en tiempo real variables clave del entorno físico, con el fin de mejorar las condiciones de aprendizaje y bienestar tanto de estudiantes como de los docentes, promoviendo entornos más saludables de estudio, inclusivos y eficientes.

2.2. Objetivos específicos

- Identificar y analizar variables ambientales relevantes que inciden en la calidad del ambiente en salas de clases, como temperatura, humedad, concentración de CO_2, ruido e iluminación, justificando su monitoreo.
- Establecer los criterios técnicos y funcionales que debe cumplir el sistema propuesto, considerando aspectos como escalabilidad, latencia, autonomía operativa, capacidad de procesamiento y tolerancia a fallos.
- Diseñar un modelo de arquitectura distribuida tipo Edge–Fog–Cloud, que represente de forma clara el funcionamiento jerárquico del sistema, la distribución de responsabilidades y el flujo de información entre sus componentes.
- Definir los mecanismos de comunicación e interoperabilidad entre sensores, nodos locales y servicios en la nube, seleccionando protocolos adecuados como MQTT y HTTPS, y políticas de sincronización periódica.
- Proponer una estructura de base de datos para el almacenamiento y análisis de los datos recolectados, considerando tanto almacenamiento local como en la nube, y los requerimientos de acceso por parte de usuarios.
- Estimar los recursos técnicos, económicos y humanos necesarios para la implementación y mantenimiento del sistema, incluyendo un análisis preliminar de costos y herramientas tecnológicas.

3. Descripción del sistema distribuido

El sistema distribuido de monitoreo ambiental en escuelas está diseñado como una solución tecnológica integral y escalable, orientada a mejorar las condiciones de confort y salubridad dentro de las salas de clases. Utiliza sensores IoT para recolectar variables críticas como temperatura, humedad, niveles de CO_2, ruido e iluminación, procesando estos datos a través de una arquitectura distribuida basada en tres capas: edge, fog y cloud. Esto permite respuestas inmediatas en el entorno físico, análisis históricos en la nube y visualización remota.

Cada aula equipada se convierte en un nodo sensorial que transmite datos en tiempo real mediante una red mallada hacia un servidor local (fog node), el cual actúa como intermediario, realizando procesamiento preliminar, emisión de alertas y sincronización posterior con la nube. El sistema está pensado para funcionar incluso en condiciones de conectividad limitada, garantizando autonomía local y tolerancia a fallos.

3.1. Arquitectura Adoptada

La arquitectura Edge–Fog–Cloud implementada distribuye jerárquicamente las tareas entre los sensores (edge), un servidor local (fog) y servicios en la nube (cloud). Esta elección permite baja latencia, operación continua incluso sin conexión a internet y alta escalabilidad horizontal.

- **Edge**: Sensores ambientales conectados a microcontroladores ESP32 capturan variables como T°, humedad, CO_2, luz y ruido.
- **Fog**: Un servidor local (Raspberry Pi 4) recibe, valida y almacena los datos en una base local PostgreSQL. Emite alertas, gestiona sensores, realiza mantenimiento predictivo y sincroniza periódicamente con la nube.
- **Cloud**: Google Cloud SQL con PostgreSQL almacena datos históricos. Grafana accede directamente a esta base para ofrecer dashboards y notificaciones. Además, la interfaz web desarrollada en Vue.js se conecta mediante API REST para permitir la gestión de alertas, umbrales y configuraciones del sistema de forma centralizada y accesible desde la nube.

3.1.1. Vía de comunicación

Para la transmisión de datos en el sistema distribuido, se emplearán distintos protocolos según el nivel de comunicación. Entre los sensores instalados en las salas se establecerá una red mallada (mesh), la cual permite que los dispositivos se comuniquen entre sí de forma descentralizada, asegurando la continuidad de la transmisión incluso ante fallos de conectividad puntuales. Esta topología es especialmente útil en entornos educativos con infraestructura limitada, ya que garantiza cobertura sin requerir un punto de acceso central[2].

[2] Jurado-Calero, R., Castillo-Montes, C., Vera Mera, M. V., & Salgado Ortiz, P. (2022). Red MESH como modelo alternativo de conectividad en instituciones de educación superior: caso de estudio Universidad Técnica Luis

La comunicación entre los sensores y el servidor local (fog node) se realizará mediante el protocolo MQTT (Message Queuing Telemetry Transport). Este protocolo ha sido ampliamente adoptado en entornos IoT por su eficiencia, bajo consumo de recursos y facilidad de implementación, cualidades que lo hacen ideal para microcontroladores como el ESP32 y para instituciones con recursos limitados[3]. Esta estructura mejora la escalabilidad y permite una transmisión eficiente con bajo uso de ancho de banda. Además, gracias a su diseño ligero, encabezados mínimos y mecanismos de reconexión automática, garantiza un funcionamiento confiable incluso en condiciones de red inestables[4].

Finalmente, la conexión entre el servidor local y la nube se implementará mediante scripts programados que utilizan el protocolo HTTPS para enviar datos en bloques. Este método permite sincronizar la información de forma segura y cifrada, y además está diseñado para tolerar conectividad limitada, reintentando el envío de datos cuando se restablece la conexión[5].

3.1.2. Base de datos

Se utiliza PostgreSQL como base de datos única y compartida entre la capa fog y la cloud, debido a que es una solución robusta, gratuita y compatible con datos estructurados. En este sistema, la instancia local se encarga de almacenar los datos de forma temporal en el fog node, mientras que la instancia en la nube guarda el histórico. Soporta series temporales y relaciones estructuradas entre entidades (sensores, salas, variables, usuarios). Grafana se conecta directamente a esta base.

3.1.3. Plataforma de Visualización

El sistema utiliza dos componentes complementarios para la visualización y gestión:

- **Grafana**: permite a usuarios autorizados acceder a dashboards interactivos desde cualquier navegador y en tiempo real. Esta herramienta facilita la exploración y análisis de los datos mediante funciones de búsqueda, filtrado y agregación de valores.[6] Los dashboards se integran directamente con PostgreSQL, y su visualización se restringe mediante autenticación por roles.

Vargas Torres de Esmeraldas. *Sapienza: International Journal of Interdisciplinary Studies*, 3(2). https://doi.org/10.51798/sijis.v3i2.314ResearchGate

[3] ¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/mqtt/

[4] Redacción. (2020, 20 de abril). MQTT Guía con toda la información que debes saber de este sistema. Descubre Arduino. https://descubrearduino.com/mqtt-que-es-como-se-puede-usar-y-como-funciona/#Como_funciona_MQTT

[5] López, M. (2022, 18 de agosto). *Protocolo HTTPS: Por qué tenerlo en tu sitio web*. IMMUNE Technology Institute. https://immune.institute/blog/protocolo-https-que-es-ventajas/

[6] Moreno, P., & Moreno, P. (2024, 7 mayo). ¿Qué es Grafana y para qué sirve? *Blog Centro de e-Learning*. https://blog.centrodeelearning.com/que-es-grafana/#que_es_grafana

- **Vue.js**: Framework progresivo de JavaScript utilizado para desarrollar una interfaz web administrativa personalizada. Esta interfaz permite a usuarios autorizados (como personal técnico o administrativo) crear, editar y desactivar umbrales de alertas, gestionar configuraciones de sensores por sala, y controlar accesos por roles. Vue.js también facilita la interacción directa con el backend vía API REST, mejorando la usabilidad frente a operaciones administrativas que Grafana no cubre de forma nativa.

Esta combinación permite ofrecer una solución robusta: por un lado, Grafana cubre la visualización analítica y en tiempo real de los datos; por el otro, Vue.js permite una interacción dinámica, editable y enfocada en la gestión del sistema, aportando flexibilidad y control personalizado al usuario final.

3.1.4. Plataforma de Visualización y Monitoreo (Desarrollo)

La plataforma de visualización fue desarrollada como una aplicación web responsiva, lo que permite que sea accesible desde distintos dispositivos, incluidos computadores, tablets y teléfonos móviles, sin necesidad de instalar una aplicación nativa. Esto facilita el despliegue y mantenimiento, ya que todas las mejoras o actualizaciones se aplican directamente en el servidor y quedan disponibles para todos los usuarios en tiempo real a través del navegador.

Aunque no se contempla una app móvil independiente, el diseño de la interfaz (implementada con Vue.js) permite un uso cómodo y funcional desde los dispositivos móviles, adaptando los paneles y componentes según el tamaño de pantalla.

Por otro lado, respecto a la visualización de datos por sala, el frontend proporciona dashboards individuales por salas, en los que se pueden consultar los datos históricos de variables como CO_2, temperatura, humedad, ruido e iluminación, junto con sus umbrales actuales. Además, los usuarios con permisos adecuados podrán modificar dinámicamente los umbrales de cada variable por sala, lo que permite un mejor ajuste del sistema de alertas.

Se optó por mostrar toda la información desde un panel central en lugar de tener dashboards separados en cada nodo porque así el sistema es más simple de manejar, se evita sobrecargar los dispositivos locales como el Raspberry Pi, y todo se puede controlar y mantener fácilmente desde un solo lugar.

3.1.5. Backend del sistema

El backend se desarrolla en Go (Golang) utilizando el framework Gin, este es elegido por su alto rendimiento, manejo eficiente de concurrencia y bajo consumo de recursos, lo que lo hace ideal para procesar datos IoT en tiempo real. Se encarga de recibir datos vía MQTT, analizarlos, validar umbrales, generar alertas y exponer una API

REST que permite gestión de usuarios, configuraciones y visualización por parte del frontend (Vue.js) o Grafana. Se implementa como un contenedor Docker independiente, conectado al contenedor de base de datos.

3.2. Componentes principales del sistema

Esta sección describe los elementos físicos, lógicos y estructurales clave que componen el sistema distribuido. Incluye desde los sensores encargados de captar las variables del ambiente de cada sala de clases hasta los nodos de procesamiento, la infraestructura de almacenamiento y su visualización.

3.2.1. Sensores y microcontroladores

El sistema considera la implementación de sensores especializados para monitorear variables ambientales clave en las salas de clases, tales como la temperatura, humedad relativa, niveles de dióxido de carbono (CO_2), intensidad del ruido y niveles de iluminación. Estas variables no solo son esenciales para mantener condiciones saludables y confortables para la comunidad educativa en general, sino que adquieren un rol aún más relevante cuando se consideran estudiantes con Trastorno del Espectro Autista (TEA), quienes pueden ser especialmente sensibles a ciertos estímulos del entorno físico.

La distribución de los sensores será estratégica, permitiendo obtener mediciones precisas de las condiciones ambientales en tiempo real. Cada nodo enviará los datos a un servidor local, y luego éste a la nube, donde serán procesados y visualizados. En caso de detectarse condiciones desfavorables como altos niveles de dióxido de carbono, ruido excesivo o iluminación inadecuada, el sistema emitirá alertas automáticas, recomendando acciones como abrir ventanas, ajustar la iluminación o reducir el volumen de ciertos dispositivos, lo que puede contribuir significativamente a crear entornos más predecibles y regulados para estudiantes con TEA.

Esta capacidad de respuesta personalizada se traduce en un aporte directo a las políticas de inclusión escolar, ya que permite adaptar los entornos educativos para minimizar factores que puedan generar sobrecarga sensorial en estudiantes neurodivergentes. La frecuencia de monitoreo y los umbrales de alerta serán configurables, permitiendo adaptar el sistema según las necesidades de cada establecimiento y población estudiantil.

Cada nodo estará equipado con un microcontrolador ESP32, reconocido por su bajo costo, versatilidad y conectividad integrada (WiFi y Bluetooth). Este microcontrolador será el encargado de gestionar sensores como:

- DHT22 para temperatura y humedad.
- MH-Z19 para el dióxido de carbono (CO_2).

- KY-038 para niveles de ruido.
- LDR para iluminación.

El software embebido será desarrollado en Golang, utilizando el framework TinyGo, que es compatible con ESP32. Esto permitirá aprovechar al máximo las capacidades del microcontrolador, facilitar la integración con los servicios backend y optimizar el consumo energético mediante modos de bajo consumo cuando los sensores no estén activos.

Además, en caso de pérdida de conectividad, los sensores estarán preparados para almacenar temporalmente los datos en la memoria interna del ESP32, garantizando la continuidad del monitoreo y evitando pérdidas de información.

En cuanto a la autonomía energética, si bien se considera la alimentación directa desde la red eléctrica del establecimiento, también se evaluará el uso de baterías recargables o paneles solares en instalaciones donde no sea posible el cableado o se requiera una solución portátil. Independientemente de la fuente de energía, el sistema contará con un plan de mantenimiento periódico para revisar el funcionamiento de los sensores, prevenir fallos y asegurar una operación continua y confiable.

3.2.2. Servidor local (Fog Node)

El servidor local, también denominado nodo fog, constituye el núcleo de procesamiento intermedio del sistema distribuido. Este dispositivo se instala físicamente dentro del establecimiento educacional, cumpliendo un rol fundamental en la arquitectura **Edge–Fog–Cloud**, al ubicarse estratégicamente entre los sensores (edge) y la infraestructura en la nube (cloud). Su propósito principal es garantizar el funcionamiento autónomo, continuo y en tiempo real del sistema, incluso ante condiciones de conectividad externa limitadas o inestables.

El servidor local se encarga de recibir, procesar y gestionar los datos ambientales recolectados por los sensores distribuidos en cada sala de clases. Estos datos llegan a través de la red mallada WiFi implementada mediante ESP-MESH. Una vez recibidos, el servidor realiza un conjunto de tareas críticas:

- **Procesamiento preliminar de datos**: Filtra y valida las lecturas provenientes de sensores, identificando valores anómalos, promediando datos y almacenando datos en una base de datos PostgreSQL para su posterior análisis y visualización.
- **Generación de alertas en tiempo real**: Evalúa continuamente los valores medidos contra umbrales predefinidos (por ejemplo, niveles de CO_2 superiores a 1.000 ppm o ruido mayor a 70 dB), generando alertas automáticas y recomendaciones para intervenir en el entorno (como ventilar la sala, ajustar

iluminación o reducir el volumen de dispositivos). Esta capacidad es especialmente importante en espacios donde estudian niños con Trastorno del Espectro Autista (TEA), ya que permite actuar rápidamente ante situaciones de sobrecarga sensorial.

- **Autonomía operativa**: En caso de pérdida de conexión con la nube, el servidor local continúa funcionando sin interrupciones. Registra los datos localmente en PostgreSQL y mantiene el sistema activo, asegurando la continuidad de la supervisión ambiental. Una vez restablecida la conexión, los datos pendientes son sincronizados con la infraestructura remota.

- **Gestión de red y dispositivos**: El servidor administra el estado de cada nodo sensor conectado a la red mallada. Es capaz de verificar la conectividad de los dispositivos, sus últimas transmisiones, niveles de batería y ejecutar actualizaciones remotas de firmware cuando se requiere modificar o mejorar el comportamiento del sistema. Esto evita la necesidad de intervención física en cada sala, facilitando el mantenimiento centralizado.

El servidor fog será una Raspberry Pi 4, elegida por su bajo consumo energético, portabilidad y capacidad para ejecutar servicios críticos del sistema[7]. El controlador Raspberry Pi 4 es también escogido debido a su bajo costo, suficiente capacidad de procesamiento y conectividad incluso en lugares de baja cobertura en relación a la competencia. Este nodo intermedio será responsable de procesar y validar los datos localmente, generar alertas en tiempo real, administrar los sensores conectados a la red, almacenar toda la información del sistema en una única base de datos PostgreSQL, y sincronizarla posteriormente con la nube. Además, podrá ejecutar actualizaciones remotas del firmware, registrar eventos operativos y mantener el sistema operativo durante desconexiones temporales, mejorando la resiliencia, escalabilidad y mantenibilidad del sistema.

3.2.3. Infraestructura en la nube (Cloud layer)

La capa de infraestructura en la nube representa el nivel más alto dentro de la arquitectura del sistema distribuido y cumple funciones clave relacionadas con la centralización, persistencia, análisis histórico y visualización remota de los datos recolectados por los sensores ambientales en cada establecimiento educacional. Esta capa permite extender el alcance funcional del sistema más allá del servidor local, facilitando la gestión multi-sede, la trazabilidad temporal y la toma de decisiones estratégicas basadas en información consolidada.

Para este sistema se opta por utilizar Google Cloud Platform (GCP) como la plataforma principal de infraestructura en la nube, específicamente mediante el servicio Google Cloud SQL con PostgreSQL como base de datos. Esta opción permite

[7] Fromaget, P. (s. f.). *Pros y contras de la Raspberry Pi: ¿Deberías comprar una?* RaspberryTips. https://raspberrytips.es/raspberry-pi-pros-y-contras/

guardar los datos históricos de manera segura, siempre disponibles y con respaldos automáticos sin más complicaciones. Además, se puede acceder a la base de datos desde los servidores locales a través de conexiones cifradas. Usar GCP también abre la posibilidad de escalar fácilmente el sistema en el futuro o integrar otras herramientas de Google si se necesitan.

La comunicación entre el servidor local (fog node) y la nube se realiza a través de sincronización programada mediante scripts cron, que ejecutan tareas automáticas en intervalos definidos (por ejemplo, cada 5, 15 o 30 minutos). Estos scripts envían bloques de datos a una instancia remota de PostgreSQL, utilizando canales seguros bajo protocolo HTTPS. Este enfoque es particularmente adecuado para entornos con conectividad intermitente, ya que permite almacenar los datos localmente y sincronizarlos de forma segura y controlada una vez que se restablece la conexión. Para mantener una correcta escalabilidad del manejo de datos y la eficiencia en el manejo de series de tiempo se considerará utilizar particionamiento y optimización tanto de manera local (fog node) como en el registro en la nube (Google Cloud SQL), considerando que el servicio puede mantenerse operativo durante muchos años, inclusive si los datos son pequeños.

4. Metodología para el desarrollo del proyecto

4.1. Metodología escogida

Para el desarrollo del proyecto se opta por una metodología con un enfoque híbrido, entre la metodología Scrum y la metodología Waterfall o cascada.

4.2. Justificación

Se utilizará una metodología híbrida rescatando los siguientes aspectos de cada una concretamente como se expresa a continuación:

- **Scrum**
  - Se desarrollan las 3 entregas del proyecto en ciclos de desarrollo relativamente cortos (3 a 4 semanas).
  - Existe una retroalimentación docente relacionada a cada entrega y una esperada posterior corrección de cada incremento en base a lo realizado. Y además, debido a la alta interacción con la docencia y el ayudante relacionado al proyecto, se facilita la detección de errores y la corrección oportuna de estos.
  - Se mantiene una priorización del trabajo en el marco del desarrollo mismo del proyecto, con la intención de mantener un eje central en la ejecución del trabajo.
- **Waterfall**
  - Se sigue una estructura lineal en el desarrollo del proyecto, concretamente se realiza un análisis y diseño general del proyecto para el primer entregable, luego se implementan y prueban aquellas funcionalidades críticas o de mayor impacto

en el proyecto para el segundo entregable y finalmente se finaliza el proyecto en su completitud para el tercer y último entregable
    - Existe una facilidad mayor en la gestión del proyecto debido a que solamente se tiene, como involucrados en los procesos de desarrollo del proyecto, al equipo de trabajo, el docente y el ayudante.
    - El proyecto no tiene un cliente principal asociado, es decir, existe un público objetivo para el cual la solución será implementada, sin embargo, no se trabaja de la mano con un cliente en concreto, lo cual puede ser importante debido a la poca flexibilidad ante la necesidad de cambios en la solución en el futuro.

4.3. Ciclo de vida del proyecto

A continuación se describirán las etapas de desarrollo del proyecto y cómo se irán recorriendo:

- **Inicio**: En esta etapa se estudia una solución para la problemática ante la cual se debe desarrollar una solución, escogiendo finalmente la alternativa más adecuada considerando y efectuando también estudios de viabilidad técnica, económica y geográfica de la misma. Se levantan los requerimientos de la solución y se plantea un eje de desarrollo consistente con el producto de software que se planea confeccionar.
- **Planificación**: Los entregables han sido previamente definidos por la docencia, por lo que se programan los hitos más importantes relacionados a cada entrega y aquellos aspectos que se deben satisfacer. Se definen las líneas base del proyecto, cuidando el alcance que tiene el proyecto como una manera de mantener la viabilidad del mismo. Además se desarrollan internamente en el equipo de trabajo debido a la ausencia de clientes directos durante el desarrollo del proyecto.
- **Ejecución**: Se desarrollan todos los aspectos técnicos del proyecto, la construcción de código y pruebas iterativas necesarias para la validación de la calidad del producto, y además verificar que los requerimientos funcionales y no funcionales se cumplen satisfactoriamente.
- **Control y seguimiento**: En el presente proyecto se consideran métricas de progreso en cantidad de paquetes de trabajo, debido a la facilidad en el manejo de futuras correcciones realizadas por docencia posterior a la entrega de cada iteración. Se manejan documentos con el afán de presentar las correcciones requeridas al equipo de trabajo a partir de cada entregable y existe la posibilidad de reuniones con docencia y/o el ayudante para resolver problemas y dudas relacionadas.
- **Cierre**: Se plantea inicialmente la finalización del proyecto con la entrega final del producto en una presentación del trabajo realizado, tanto al docente como a posibles clientes interesados en la solución desarrollada.

4.4. Roles y responsabilidades

Se distribuirán los roles y responsabilidades en el equipo de trabajo como sigue en la siguiente:

- **Project Manager**: Javiera Vergara
- **Product Owner**: Alexandra Navarro
- **Scrum Master**: Ariel Núñez
- **Desarrollo Back-End**: Javiera Vergara
- **Desarrollo Front-End**: Alexandra Navarro
- **Analista Funcional**: Ariel Núñez

Independiente de lo establecido en la repartición de roles, se establece que todos los miembros del equipo son polifuncionales, dado que al ser solamente tres personas desarrollando todo el proyecto, posiblemente será necesario que alguno de los integrantes apoye en alguna de sus funciones a otro.

4.5. Control de Calidad

Para asegurar la calidad del sistema desarrollado y su correcto funcionamiento en un entorno simulado, se aplican las siguientes prácticas:

- **Pruebas manuales**: Se realizan pruebas manuales de cada uno de los componentes principales del sistema (backend, frontend, base de datos y visualización), utilizando los datos generados por el simulador de sensores. Esto permite verificar que:
  - Los datos generados por el simulador son correctamente recibidos por el backend.
  - Los datos se almacenan correctamente en la base
  - Las alertas se generan de acuerdo con los umbrales configurados.
  - La interfaz muestra en tiempo real los datos y alertas asociados a cada sala.
  - Los umbrales pueden editarse y su modificación se refleja en el comportamiento del sistema
- **Dashboards**: Se utilizaron dashboards en Grafana para validar visualmente el flujo completo de los datos desde el sensor (simulado) hasta su visualización. También se verificó la trazabilidad de los datos mediante la consulta de la base de datos y el historial de alertas.

Sin embargo, dado que en esta etapa del desarrollo aún no se dispone de sensores físicos reales, no se ha considerado la ejecución de pruebas de integración hardware–software en un entorno real. Este tipo de validación queda planificada para una futura fase de implementación, una vez que el sistema pueda operar con dispositivos físicos en condiciones reales de uso.

5. Herramientas de trabajo e implementación

5.1. Herramientas organizacionales

Para poder hacer efectivo este trabajo, se considera que es fundamental utilizar herramientas de trabajo organizacional como:

- **Google Drive**: se utiliza como plataforma de almacenamiento centralizada para todos los archivos del proyecto, incluyendo informes, diagramas, documentación técnica y registros de avance. Gracias a su sistema de carpetas compartidas y control de permisos, permite mantener una organización clara, acceso seguro desde múltiples dispositivos y colaboración eficiente entre los integrantes del equipo.

- **Google Docs:** se emplea como herramienta principal de redacción colaborativa, permitiendo a los miembros del grupo trabajar simultáneamente sobre los informes y documentos técnicos. Su funcionalidad de comentarios, sugerencias y control de versiones facilita la revisión continua, la integración de observaciones externas y la elaboración iterativa de los entregables.

- **GitHub:** se utiliza como sistema de control de versiones y repositorio de código fuente. Permite llevar un seguimiento más detallado de los cambios que se van realizando en el software, fomenta la colaboración, administra ramas de desarrollo y facilita la integración continua. Además se usa para documentar el código, registrar incidencias y organizar tanto tareas como proyectos.

- **Discord:** utilizado como canal de comunicación principal entre los integrantes del equipo. Su sistema de canales, mensajería instantánea y llamadas de voz permite una comunicación mucho más fluida y constante durante todo el proyecto. También se utiliza para realizar reuniones, coordinar entregas y resolver dudas en tiempo real.

5.2. Herramientas de implementación

El desarrollo del sistema distribuido de monitoreo ambiental se apoya en un conjunto de herramientas, lenguajes y frameworks que permiten construir una solución modular, escalable y mantenible. Todos los componentes están desplegados como contenedores Docker, lo que asegura portabilidad, facilidad de pruebas y consistencia entre entornos locales y en la nube.

- **Backend:**
    - **Lenguaje de programación:** Go (Golang), elegido por su rendimiento, concurrencia nativa y eficiencia en el manejo de múltiples sensores y procesos simultáneos.
    - **Framework web:** Gin, utilizado para desarrollar una API REST que permite consultar datos, administrar sensores y umbrales, y servir al frontend.
    - **Contenedor Docker:** El backend se despliega en un contenedor que expone servicios REST y gestiona la lógica del sistema distribuido.
- **Base de datos:**

- **Sistema de gestión:** PostgreSQL, tanto local (fog node) como en la nube (Google Cloud SQL).
- **Contenedor Docker:** En desarrollo local se utiliza un contenedor PostgreSQL que almacena series temporales y la estructura relacional (escuelas, salas, sensores, variables, usuarios, alertas).

- **Frontend:**
  - **Framework principal:** Vue.js, utilizado para crear una interfaz web administrativa. Esta interfaz permite configurar umbrales personalizados, visualizar alertas por sala y gestionar el sistema por roles.
  - **Framework de diseño:** Vuetify, usado para construir componentes visuales modernos y accesibles.
  - **Contenedor Docker:** El frontend se ejecuta como un contenedor independiente y se comunica con el backend a través de API REST.
- **Visualización**
  - **Plataforma:** Grafana, conectada directamente a PostgreSQL. Permite monitoreo en tiempo real, visualización de históricos por sala, dashboards interactivos y alertas visuales o por correo.
  - **Contenedor Docker:** Grafana se ejecuta en su propio contenedor y accede a la base de datos mediante credenciales seguras.
- **Microcontroladores:**
  - ESP32, por su bajo costo, conectividad WiFi/Bluetooth integrada y soporte en entornos embebidos.
- **Sensores:**
  - **DHT22**: Medición de temperatura y humedad.
  - **MH-Z19**: Medición de niveles de CO_2.
  - **KY-038**: Medición de niveles de ruido.
  - **LDR**: Medición de intensidad lumínica.
- **Comunicación:**
  - **Protocolo MQTT**: Utilizado entre sensores (ESP32) y el servidor local para transmisión eficiente y en tiempo real de datos mediante el modelo de publicación/suscripción.

- **Red mallada (ESP-MESH)**: Implementada entre nodos ESP32 para asegurar conectividad continua y resiliente incluso en espacios amplios o con interferencias estructurales.
- **Sincronización fog–cloud**: Se realizará mediante scripts programados (cron) que envían bloques de datos desde el servidor local hacia PostgreSQL en la nube utilizando protocolos seguros como HTTPS.

- **Hardware servidor local:**
  - **Hardware**: Raspberry Pi 4, usado como nodo fog. Procesa los datos, genera alertas locales, administra sensores, almacena temporalmente en PostgreSQL local y sincroniza con la nube.
- **Seguridad:**
  - **Cifrado TLS**: Utilizado en las comunicaciones hacia la nube para proteger la integridad y confidencialidad de los datos.
  - **Autenticación por roles**: Definida para controlar el acceso a los dashboards y funciones críticas del sistema, especialmente en la capa cloud.
- **Nube:**
  - **Google Cloud Platform (GCP):** utilizada como plataforma principal para alojar la instancia remota de PostgreSQL (Google Cloud SQL). Permite sincronizar datos desde los servidores fog mediante conexiones seguras, gestiona respaldos automáticos, aplica control de acceso por roles y escalar el sistema en futuras fases.
- **Simulador de sensores:** En esta etapa del desarrollo, como aún no se cuenta con sensores físicos reales, el sistema incorpora un simulador que genera datos ambientales ficticios. Este componente, desarrollado en Go y ejecutado como un contenedor Docker, reproduce el comportamiento de los sensores ESP32, enviando datos periódicos de temperatura, CO_2, humedad, ruido e iluminación al backend mediante el protocolo MQTT.
  - **Lenguaje:** Go (Golang), empleado para simular la emisión periódica de datos ambientales (T°, CO_2, humedad, ruido, luz)
  - **Contenedor Docker:** Un contenedor específico simula sensores publicando datos vía MQTT, permitiendo pruebas de funcionamiento del sistema en entornos controlados.

  Gracias a este simulador, es posible validar el flujo completo del sistema, desde la recepción y almacenamiento de datos hasta la visualización y generación de alertas en las salas, sin necesidad de contar con hardware físico, permitiendo pruebas controladas y asegurando el correcto funcionamiento de la lógica del sistema.

6. Modelados y diseños del sistema

6.1. Diagrama de Caso de Uso

Figura 1: Diagrama de casos de uso.

6.2. Diagrama de Arquitectura

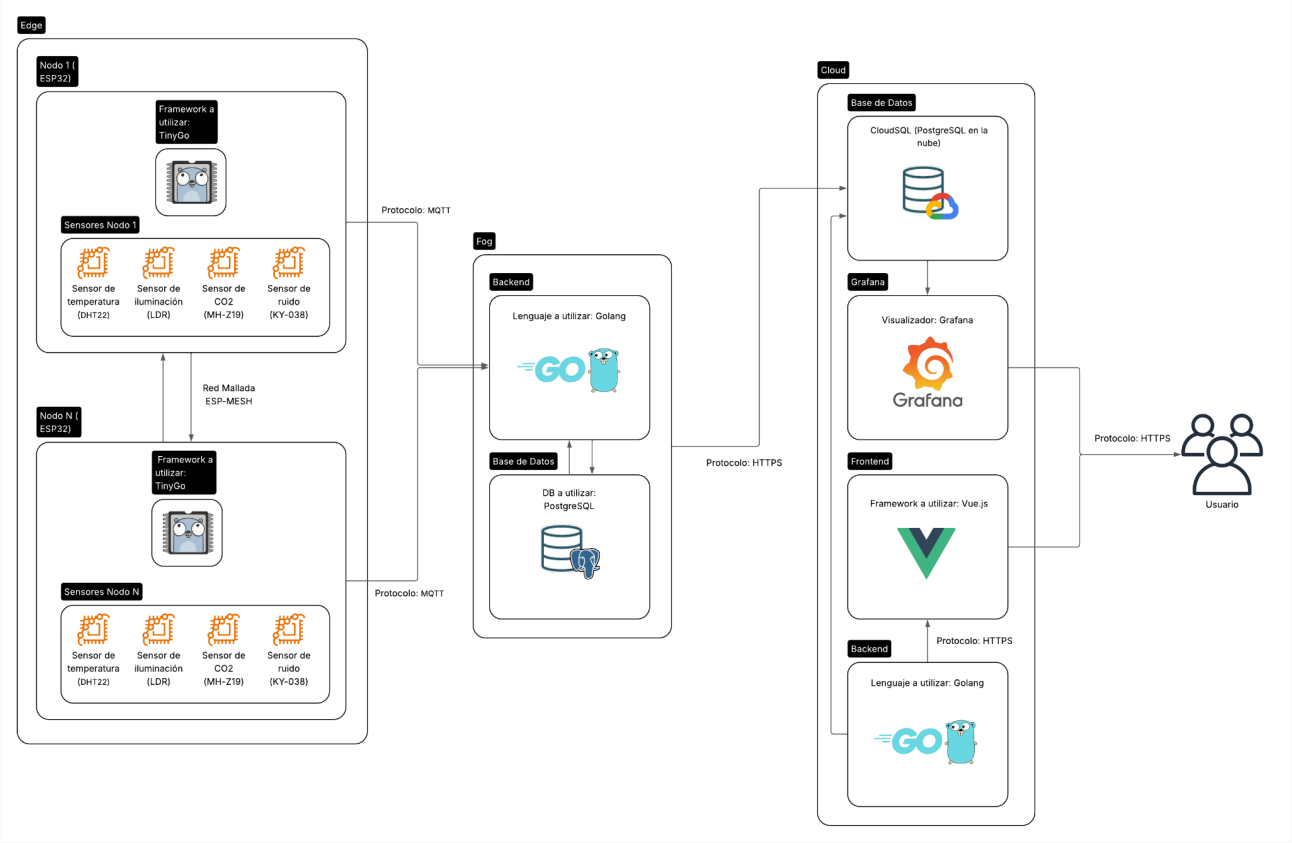

Figura 2: Diagrama de arquitectura.

6.3. Diagrama de Proceso Actual

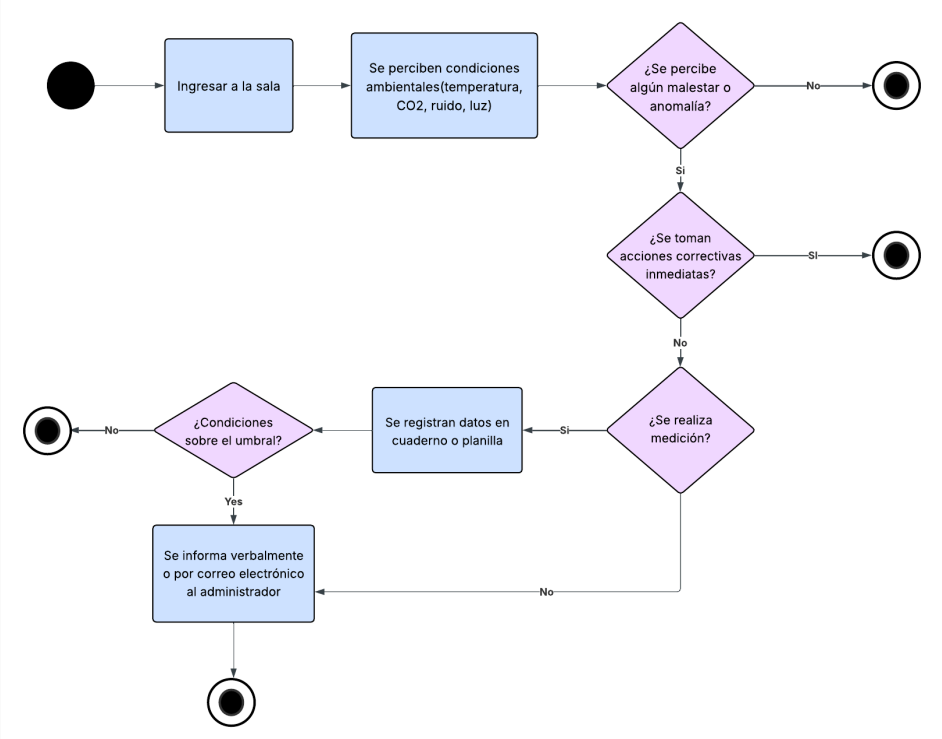

Figura 3: Diagrama de procesos sin solución.

6.4. Diagrama de Proceso con la solución.

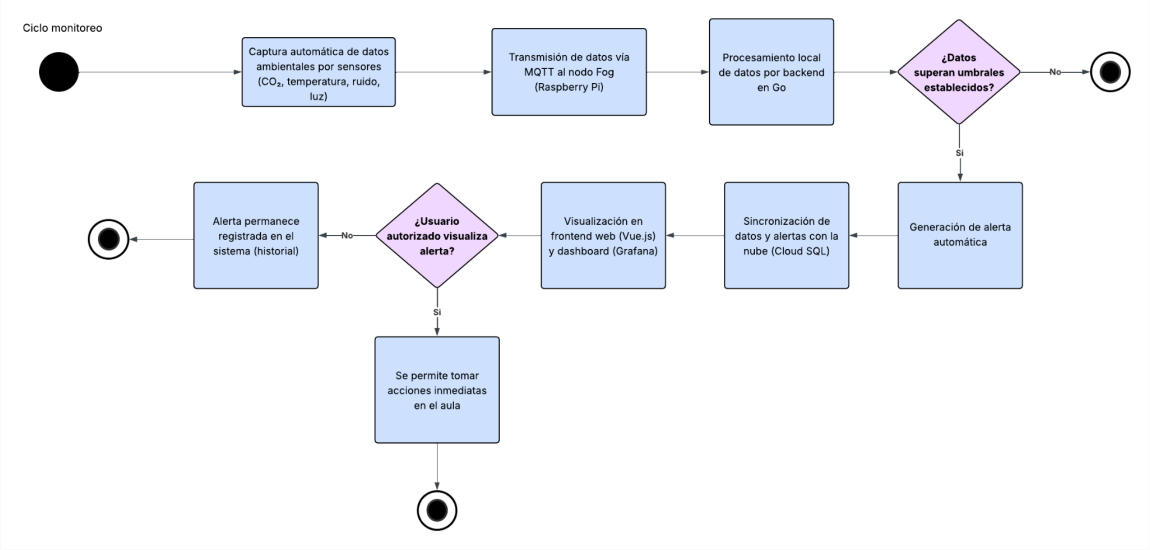

Figura 4: Diagrama de procesos con la solución implementada.

7. Base de datos y modelado de datos

Para la gestión de datos, se ha optado por utilizar **PostgreSQL v16** como sistema de base de datos, debido a su robustez, flexibilidad y compatibilidad con entornos distribuidos. Esta tecnología es ideal para estructurar relaciones entre entidades clave del sistema (como sensores, salas, variables y mediciones) mediante el uso de claves primarias y foráneas, lo que facilita la integridad referencial y la trazabilidad de la información. Además, PostgreSQL permite el almacenamiento eficiente de series temporales mediante tipos de datos como timestamp, junto con funciones de agregación y filtrado que optimizan el análisis ambiental, favoreciendo una respuesta oportuna ante condiciones críticas dentro del entorno escolar.

7.1. Estructura de la base de datos

7.1.1. Diagrama representativo

Esta sección presenta los esquemas visuales utilizados para modelar la estructura lógica de la base de datos del sistema. Se incluyen dos niveles de abstracción: el **Modelo Entidad-Relación (MER)**, que permite identificar las entidades, atributos y relaciones conceptuales del sistema; y el **Modelo Relacional (MR)**, que traduce dicha estructura conceptual a un formato compatible con la implementación en PostgreSQL. Ambos modelos contribuyen a garantizar la integridad de los datos, la eficiencia del almacenamiento y la claridad en el diseño del sistema de información.

7.1.1.1. Modelo Entidad-Relación (MER)

A continuación, se presenta el **Modelo Entidad-Relación (MER)** del sistema, el cual representa de manera conceptual las principales entidades involucradas, sus atributos clave y las relaciones entre ellas. Este diagrama permite visualizar la estructura lógica del sistema antes de su implementación física, facilitando la comprensión de cómo se organizan y conectan los distintos componentes de la base de datos.

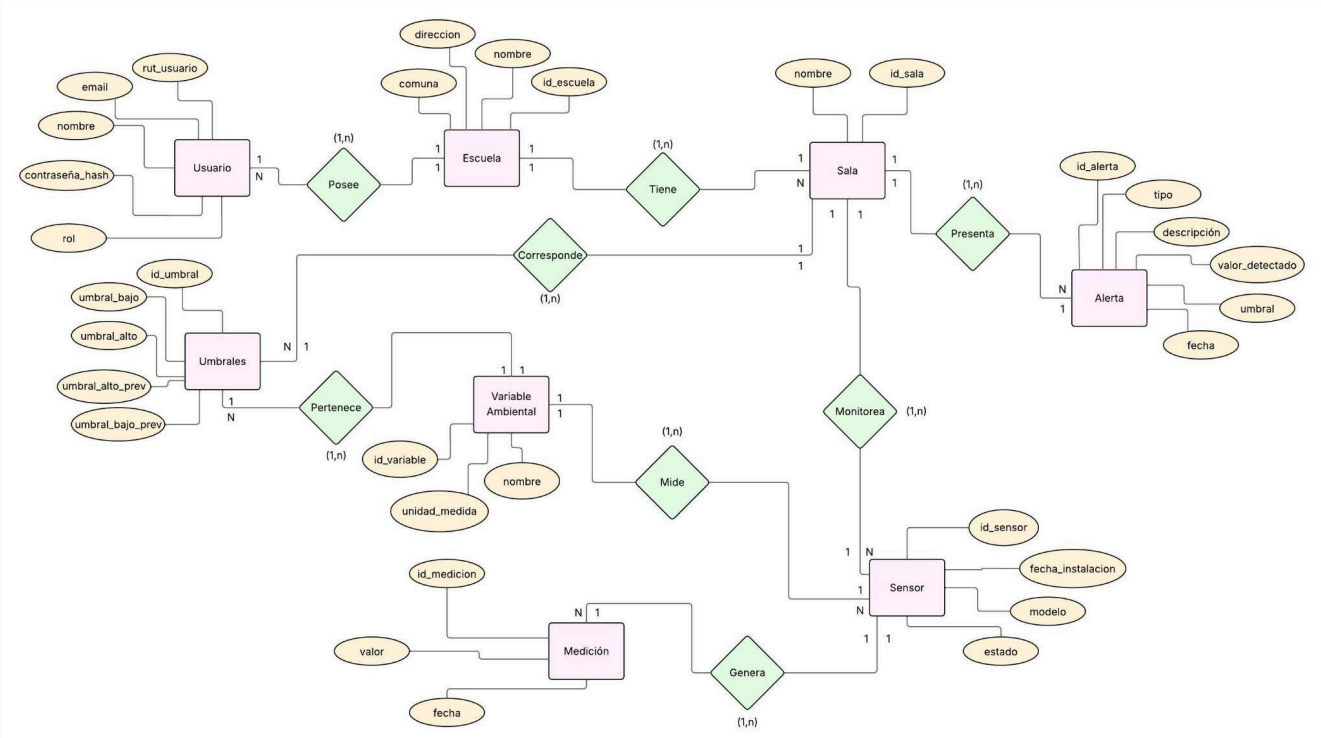

Figura 5: Modelo Entidad-Relación (MER).

7.1.1.2. Modelo Relacional (MR)

El siguiente diagrama muestra el **Modelo Relacional (MR)**, derivado del MER, donde las entidades se representan como tablas con sus respectivas claves primarias y foráneas. Este modelo refleja la estructura que será implementada en PostgreSQL y sirve como base para la gestión de los datos estructurados y de series temporales dentro del sistema de monitoreo ambiental.

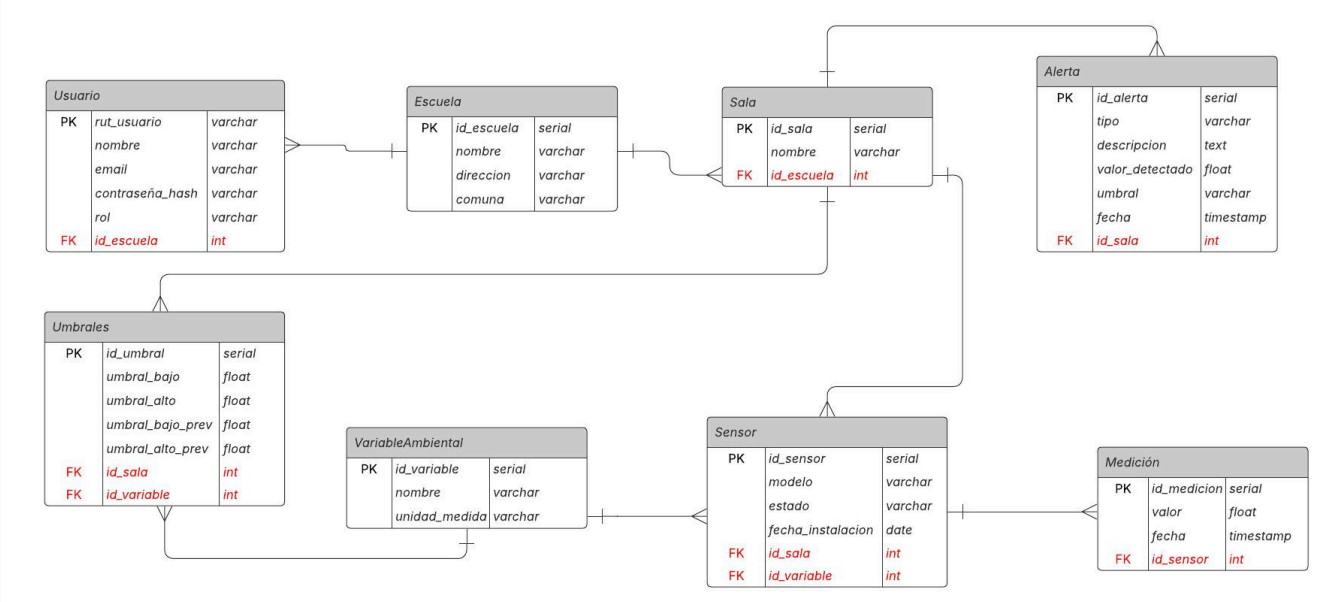

Figura 6: Modelo Relacional (MR).

En esta versión del modelo relacional se incorpora la tabla *Umbrales*, cuyo propósito es permitir la configuración de umbrales personalizados por variable y por sala. Esta decisión surge para permitir que en cada sala se puedan ajustar los rangos de alerta según sus propias condiciones ambientales o necesidades pedagógicas. Al agregar esta tabla, el modelo de la base de datos queda mejor organizado y ahora es mucho más fácil manejar y editar los rangos de alerta directamente desde la interfaz web.

7.1.2. Diccionario de datos

El diccionario de datos se encuentra anexo en un excel. Este describe en detalle cada tabla de la base de datos, indicando sus atributos, tipos de datos, claves y descripciones.

8. Escalabilidad y manejo de Series Temporales

Uno de los desafíos clave para este sistema de monitoreo corresponde al manejo eficiente de series temporales, ya que el sistema puede crecer rápidamente en volúmen de datos generados por los sensores ambientales. El modelo de datos fue diseñado considerando que cada sensor emite datos periódicamente, lo cual genera una secuencia temporal de registros que puede crecer de forma sostenida a lo largo del tiempo.

Para abordar este problema, se opta por lo siguiente:

- **Estructura Optimizada para Series de Tiempo:** Las mediciones se almacenan en una tabla llamada ‘*medición’,* normalizada y relacionada mediante claves foráneas con entidades como *‘sensor’, ‘variable’, ‘sala’ y ‘fecha’*. Esto permite la utilización de índices compuestos y consultas parametrizadas por fecha y ubicación, lo cual garantiza un acceso eficiente a los datos relevantes.

- **Consultas acotadas y visualización eficiente:** Tanto el backend como la visualización del sistema (Grafana y Vue.js) están diseñados para operar con consultas acotadas por rango de fechas y por sala. Esto permite que el sistema consulte solo los datos necesarios, sin tener que revisar toda la base de datos cada vez, lo que hace que las respuestas sean rápidas y constantes, incluso cuando ya se han acumulado muchos datos en el tiempo.
- **Particionamiento en el nodo Fog:** En el nodo Fog, se usa una base de datos PostgreSQL que guarda temporalmente la información que llega desde los sensores. Para que esta base funcione bien aunque los datos vayan creciendo con el tiempo, se puede aplicar una técnica llamada particionamiento por tiempo. Esto significa dividir la tabla de mediciones en partes más pequeñas, como por meses. Gracias a esto, las búsquedas y los registros nuevos son más rápidos, ya que la base de datos no tiene que revisar toda la información cada vez, sino solo la parte correspondiente al mes que interesa.

  Además, este método permite limpiar automáticamente los datos más antiguos si así se decide, lo que ayuda a ahorrar espacio en la Raspberry, que tiene almacenamiento limitado, incluso hace más fácil la sincronización con la nube, ya que se puede identificar qué bloques de datos son nuevos o han cambiado, facilitando su envío sin tener que revisar todo desde cero.

- **Escalabilidad futura con TimescaleDB:** En etapas posteriores a las que se realizarán para este proyecto, se puede considerar la opción de TimescaleDB, una extensión de PosgreSQL especialmente optimizada para series temporales.Esta solución permite reducir el espacio que ocupan los datos antiguos comprimiendolos automáticamente, agrupar la información por tramos de tiempo, y trabajar con grandes volúmenes de datos sin que el sistema se vuelva lento ni consuma demasiados recursos.[8]

9. Políticas de diseño y supuestos

9.1. Supuestos del sistema

1. Se asume que el sistema se implementará inicialmente en 5 a 10 salas de clases por establecimiento, con posibilidad de escalado.

2. Se considera que los usuarios principales serán docentes, personal administrativo y técnico.

3. Se asume la existencia de tomas de corrientes en cada una de las salas para brindar energía a los sensores y servidores fog.

[8] Timescale. (s.f.). Documentation | Timescale. Recuperado el 12 de junio de 2025, de https://docs.timescale.com/

4. Cada establecimiento cuenta con una conectividad WiFi básica, suficiente para el funcionamiento de la red mesh interna y la sincronización eventual de la nube.
5. Los sensores capturan y transmiten datos cada 15 minutos por defecto, con la posibilidad de ajustar este tiempo según requerimientos de monitoreo.
6. Se considera que los sensores deben operar en rangos de temperatura entre 0°C y 45°C, con protección básica contra polvo y humedad (uso en interiores).
7. El servidor local (Raspberry Pi 4) será capaz de almacenar al menos una semana de datos sin conexión a la nube.
8. En caso de desconexión con la nube, se asume que el nodo fog seguirá funcionando de forma autónoma y almacenará los datos localmente.
9. Se supone que PostgreSQL en la nube tendrá la capacidad suficiente para guardar el histórico completo de datos de los sensores por al menos 2 años.
10. La configuración inicial de umbrales y dispositivos será realizada por el equipo técnico en cada institución antes de entregar el sistema.
11. Los umbrales para variables ambientales podrán ser personalizados localmente por el personal administrativo para cada sala.
12. Se asume que los datos ambientales no están vinculados a información personal, pero se mantendrán medidas de seguridad (TLS y roles de acceso).
13. Se considera que los docentes recibirán una breve inducción para interpretar los dashboards y actuar frente a alertas.
14. Se prevé una inspección mensual de sensores para calibración, limpieza y revisión del estado de batería o fuente de poder.
15. El sistema usará tecnologías abiertas como MQTT, ESP-MESH, PostgreSQL, Go, Grafana y Vue.js, lo que asegura interoperabilidad futura y evita dependencia de proveedores especializados.
16. Se supone que los ESP32 pueden ser actualizados remotamente desde el nodo fog sin necesidad de intervención física en cada sala.
17. Se estima que el software recibirá mejoras cada semestre, considerando retroalimentación de usuarios y necesidades operativas.
18. Se asume que cada Raspberry Pi 4 podrá gestionar hasta 50 sensores simultáneamente sin comprometer la estabilidad del sistema.
19. El sistema contará con un registro de logs de errores tanto en sensores como en el nodo fog, permitiendo la trazabilidad en fallos y en diagnósticos de mantenimiento.

20. El sistema está preparado para implementar un módulo de exportación de datos a CSV o PDF desde Grafana en futuras etapas del proyecto.

21. Las alertas están configuradas para activarse mediante dos niveles de criticidad: preventivo y crítico, cada uno con distinto tipo de visualización y frecuencia. Estos vienen inicializados con los siguientes valores:

    - **CO_2 (ppm):**
        - **Informativo:** 400 - 1.000 ppm (Buena calidad del aire)[9].
        - **Preventivo:** 1.001 - 1.500 ppm (Puede generar somnolencia, mala calidad del aire)[10].
        - **Crítico:** > 1.500 ppm (Dolores de cabeza, fatiga, falta de concentración)[11].
    - **Temperatura (C°):**
        - **Informativo:** 20 - 24°C (Zona de confort térmico)[12].
        - **Preventivo:** 25 - 27°C (Condiciones ligeramente cálidas)[13].
        - **Crítico:** < 20°C o > 27°C (Condiciones fuera del rango de confort térmico, con posible impacto en la salud o concentración)[14].
    - **Humedad Relativa (%):**
        - **Informativo:** 40 - 60% (Nivel óptimo para confort y salud)[15].
        - **Preventivo:** 30 - 39% o 61 - 70% (Aumenta el riesgo de virus o moho)[16].

[9] American Society of Heating, Refrigerating and Air-Conditioning Engineers. (2019). *ANSI/ASHRAE Standard 62.1-2019: Ventilation for acceptable indoor air quality*. ASHRAE.
[10] Health and Safety Executive. (2021). *Ventilation and air quality in indoor workplaces*. https://www.hse.gov.uk/coronavirus/equipment-and-machinery/air-conditioning-and-ventilation/index.htm
[11] United States Environmental Protection Agency. (2023). *Carbon dioxide (CO_2) and indoor air quality*. https://www.epa.gov/indoor-air-quality-iaq/carbon-dioxide
[12] American Society of Heating, Refrigerating and Air-Conditioning Engineers. (2020). *ANSI/ASHRAE Standard 55-2020: Thermal environmental conditions for human occupancy*. ASHRAE.
[13] World Health Organization. (2018). *WHO housing and health guidelines*. World Health Organization. https://www.who.int/publications/i/item/9789241550376
[14] Health and Safety Executive. (2021). *Thermal comfort*. https://www.hse.gov.uk/temperature/thermal/index.htm
[15] American Society of Heating, Refrigerating and Air-Conditioning Engineers. (2016). *Indoor air quality guide: Best practices for design, construction and commissioning*. ASHRAE.
[16] United States Environmental Protection Agency. (2023). *Moisture control guidance for building design, construction and maintenance*. https://www.epa.gov/indoor-air-quality-iaq

- **Crítico:** < 30% o > 70% (Favorece proliferación de patógenos, deshidratación y malestar general)[17].

- **Ruido Ambiental (dB[A]):**
  - **Informativo:** < 35 dB (Condiciones ideales para concentración)[18].
  - **Preventivo:** 36 - 55 dB (Molestias leves, posible afectación de la comunicación)[19].
  - **Crítico:** > 55 dB (Interfiere en el aprendizaje y puede generar estrés)[20].

- **Iluminación General (lux):**
  - **Informativo:** 500-750 lux (Iluminación adecuada para lectura y escritura)
  - **Preventivo:** 350 – 499 lux o 751 – 1.000 lux (Condiciones de luz subóptimas o excesivas).
  - **Crítico:** < 350 lux o > 1.000 lux (Iluminación deficiente o deslumbrante, afecta la visión y concentración)[21].

Se hace mención a los parámetros “informativos”, sólo con el fin de que se tenga conocimiento de aquellos.

22. Se considera que todos los componentes electrónicos son alimentados por corriente de 5v y estarán protegidos por fusibles o convertidores de voltajes estables.

23. El sistema permitirá agregar nuevas variables ambientales en el futuro, si se conectan sensores adicionales compatibles con ESP32.

24. Las bases de datos de PostgreSQL tanto locales como en la nube utilizarán respaldos automáticos semanales, almacenados en dispositivos externos o nube.

25. Los sensores podrán contar con baterías recargables como respaldo en caso de cortes de energía, con autonomía de al menos 2 horas.

---

[17] World Health Organization. (2009). *WHO guidelines for indoor air quality: Dampness and mould*. https://www.who.int/publications/i/item/9789289041683

[18] World Health Organization. (1999). *Guidelines for community noise*. https://apps.who.int/iris/handle/10665/66217

[19] United States Environmental Protection Agency. (1974). *Information on levels of environmental noise requisite to protect public health and welfare with an adequate margin of safety*. https://nepis.epa.gov/Exe/ZyPDF.cgi/2000L3LN.PDF

[20] Shield, B., & Dockrell, J. E. (2008). The effects of environmental and classroom noise on the academic attainments of primary school children. *Journal of the Acoustical Society of America, 123*(1), 133–144. https://doi.org/10.1121/1.2812596

[21] Ledbox. (s.f.). *Niveles recomendados de iluminación (lux) según el tipo de estancia*. Ledbox Blog. Recuperado el 12 de junio de 2025, de https://blog.ledbox.es/niveles-recomendados-lux/

26. En caso de detección de fallas críticas del sistema (ej. sensor fuera de línea), se emitirá una alerta inmediata al personal técnico.
27. El sistema será modular por lo que permitirá reemplazar sensores sin interrumpir la operación de los demás dispositivos.
28. Se asume que los nodos fog (Raspberry Pi) cuentan con reloj en tiempo real (RTC) para mantener la sincronización horaria sin conexión a internet.
29. Las instalaciones estarán protegidas contra manipulaciones físicas por parte de estudiantes (uso de carcasas o ubicaciones seguras).
30. Se contempla la posibilidad de realizar simulaciones con datos históricos para análisis preventivos y decisiones pedagógicas.
31. Las variables ambientales podrán ser visualizadas tanto en tiempo real como en gráficos históricos por hora, día o semana.
32. Se estima que el tráfico de red generado por cada sensor no excederá los 50 KB por transmisión manteniendo eficiencia en entornos con WiFi limitado.
33. Se asume que el sistema puede ser replicado en distintas escuelas con configuraciones similares sin necesidad de modificar el backend.
34. El sistema permitirá configurar alarmas silenciosas (solo visuales) en entornos donde el ruido adicional pueda ser perturbador.
35. La visualización de los dashboards estará restringida por autenticación de usuarios, evitando accesos indebidos.
36. Se asume que el sistema será instalado y mantenido por personal técnico capacitado, sin requerir habilidades avanzadas en programación por parte del establecimiento
37. Se supone que las instituciones educativas podrán solicitar soporte externo si detectan eventos inusuales o fallos persistentes en el sistema.
38. La solución será suficientemente flexible para operar en escuelas urbanas y rurales, con adaptaciones en conectividad y energía.

9.2. Diseño y política de recursos

9.2.1. Planificación y Gestión del Talento Humano

- **Equipo de desarrollo**: El sistema será desarrollado e implementado por un equipo multidisciplinario que considera las siguientes funciones claves:

- **Desarrolladores Back-end:** Están encargados de la lógica del sistema, creación de APIs RESTful en Go, conexión con PostgreSQL, y procesamiento de los datos ambientales obtenidos de los sensores.
- **Desarrolladores Front-end / Visualización:** Responsables del desarrollo de la interfaz web del sistema utilizando Vue.js, la cual permite la gestión de alertas, configuración de umbrales y visualización personalizada por roles. Además, se encargan de la configuración y personalización de dashboards en Grafana, asegurando que tanto la visualización analítica como la interacción administrativa sean claras, accesibles y útiles para docentes, personal técnico y administrativo.
- **Especialistas IoT:** Se encargan de la instalación, configuración y mantenimiento de los sensores (ESP32, DHT22, MH-Z19, etc.), así como de la red mallada (ESP-MESH) y la correcta alimentación energética.
- **Administradores de Sistemas:** Responsables de la configuración del servidor local (Raspberry Pi 4), el despliegue de la base de datos PostgreSQL y la gestión de las políticas de respaldo, monitoreo y conectividad con la nube.
- **Personal de Soporte Técnico**: Encargados de atender incidencias del hardware o software, supervisar la operación del sistema, realizar tareas de mantenimiento predictivo y brindar apoyo directo a los usuarios dentro del establecimiento

- **Capacitación:** Los usuarios finales (docentes, personal técnico y administrativo) recibirán capacitación inicial para comprender el funcionamiento del sistema, interpretar correctamente los dashboards y actuar ante los distintos niveles de alerta. Las capacitaciones se repetirán de forma anual, y adicionalmente cada vez que se realicen actualizaciones significativas del sistema o se integren nuevas funcionalidades.
- **Gestión del Talento:** El proyecto contempla una política de formación y perfeccionamiento continua dirigida al equipo técnico y de desarrollo, con el objetivo de mantener altos estándares de calidad en todas las etapas del sistema. Se incentivará la capacitación en tecnologías embebidas e IoT, plataformas de visualización como Grafana, mantenimiento predictivo, normativas de protección de datos, así como en el uso y actualización de herramientas clave como Go y PostgreSQL. Esta estrategia busca fortalecer las competencias del equipo, garantizar la sostenibilidad técnica del sistema a largo plazo y fomentar la adopción de buenas prácticas en todo el ciclo de vida del software.

9.2.2. Diseño de Infraestructura Técnica

- **Infraestructura Local y en la Nube:**
    - Cada establecimiento contará con una Raspberry Pi 4 como servidor local (fog node), que funcionará como centro de acopio, análisis y administración de los datos en tiempo real.
    - La infraestructura en la nube estará basada en Google Cloud SQL con PostgreSQL en Google Cloud Platform (GCP), lo que permitirá almacenar datos históricos, acceder de forma segura desde los servidores locales y escalar el sistema fácilmente a múltiples establecimientos.
- **Almacenamiento y red:**
    - Cada nodo fog podrá almacenar al menos 7 días de datos sin conexión a internet.
    - Los respaldos automáticos serán ejecutados semanalmente y replicados tanto localmente como en la nube.
    - La comunicación con los sensores será a través de ESP-MESH + MQTT, y la sincronización con la nube por medio de cron + HTTPS.

9.2.3. Política de Mantenimiento y Actualizaciones

- **Mantenimiento Preventivo Mensual:** Incluye limpieza, inspección física, calibración de sensores, revisión de estado de baterías y pruebas operativas de conexión.
- **Mantenimiento Correctivo:** Se ejecutará según alertas del sistema o fallos detectados por los logs automáticos.
- **Actualizaciones del Software:** Se realizarán semestralmente, incluyendo mejoras de rendimiento, nuevas funciones o ajustes de compatibilidad.
- **Pruebas de Regresión:** Antes de cada actualización, se harán pruebas locales en entornos de staging, garantizando que las funcionalidades actuales no se vean afectadas.

9.2.4. Política de Recursos Financieros

9.2.4.1. Presupuestos de Operaciones

- **Costos de Hardware:** Se contemplan los costos iniciales y de reemplazo de ESP32, sensores ambientales, Raspberry Pi, baterías 18650, módulos TP4056 y adaptadores de 5 V.

- **Infraestructura y Software:** Se destinará presupuesto al mantenimiento del nodo fog, conectividad a internet, espacio físico seguro para instalaciones, y software (Go, TinyGo, Grafana, Vue.js).
- **Mantenimiento y Soporte:** Incluye servicios técnicos, revisión de sensores, ajustes de red y gastos de inspección mensual.
- **Capacitación Continua:** Se asignan recursos a capacitaciones técnicas y pedagógicas orientadas a docentes y personal del establecimiento.
- **Contingencias y Repuestos:** Se reservará un fondo para responder a fallos críticos, vandalismo, robo o desastres naturales.

9.2.4.2. Sostenibilidad de costos

- **Evaluación de Costo-Beneficio:** Toda compra o expansión será evaluada en función del impacto en el bienestar escolar y la eficiencia del sistema.
- **Optimización de Recursos:** Se promoverá el uso compartido de dashboards y equipos en aulas cercanas si los sensores lo permiten.
- **Negociación con Proveedores:** Se buscarán acuerdos con distribuidores mayoristas de sensores y microcontroladores, y se priorizarán soluciones de código abierto.
- **Revisión Periódica de Costos:** Cada semestre se revisará la operación para identificar gastos optimizables y reducir costos sin afectar la funcionalidad.
- **Búsqueda de Financiamiento Externo:** Se explorarán oportunidades con el Ministerio de Educación, fondos de innovación o alianzas con universidades e instituciones tecnológicas.

9.2.5. Sostenibilidad Ambiental y Energética

- **Alimentación Eléctrica Mixta:** Cada sensor podrá conectarse a la red eléctrica de la sala, pero contará con baterías de respaldo (18650) y posibilidad futura de integrar pequeños paneles solares.
- **Materiales Reutilizables:** Se utilizarán carcasas protectoras impresas en 3D con materiales reciclables para alojar los sensores y microcontroladores.
- **Bajo Consumo Energético**: Todos los componentes del sistema operan en 5 V con un consumo estimado inferior a 300 mA por sensor.

9.2.6. Política de privacidad de datos

- **Cumplimiento Legal:** Se garantiza el cumplimiento de la Ley 19.628 sobre protección de datos personales y otras normativas aplicables en el contexto escolar.
- **Seguridad de los Datos:** Toda la comunicación entre nodos se cifrará mediante TLS. Se aplicarán políticas de autenticación con control por roles (docente, técnico, administrador).
- **Transparencia y Control de Datos:** Los usuarios podrán revisar su actividad, datos almacenados y el propósito de los registros ambientales. Se asegura la no vinculación con datos personales del alumnado.

10. Contenedores y despliegue

10.1. Contenedores Docker

El sistema será desplegado utilizando contenedores Docker, ya que es una solución que permite encapsular cada componente del sistema en entornos independientes y portables. Esto permite estandarizar el entorno de ejecución, evitando problemas de compatibilidad entre distintos equipos y sistemas operativos[22]. Los contenedores incluirán tanto el backend, la base de datos, el frontend, el visualizador como el simulador de sensores.

El despliegue se realizará a través de archivos docker-compose.yml, lo que facilitará la orquestación de los servicios, permitiendo iniciar, detener y actualizar múltiples contenedores a través de un solo comando. Cada nodo local ejecutará su propio conjunto de servicios en contenedores y podrá sincronizarse con servicios remotos en la nube como Google Cloud SQL.

10.1.1. Contenedor Backend

- **Lenguaje y framework:** Go (Golang) con el framework Gin.
- **Funcionalidad:** Procesa los datos provenientes de los sensores, aplica la lógica de negocio, evalúa umbrales definidos por los usuarios y genera alertas. Expone una API REST que permite la consulta de datos por parte del frontend y el sistema de visualización.
- **Componentes claves:**
    - Recepción de datos vía MQTT desde los nodos sensores (ESP32).
    - Análisis y validación de mediciones ambientales.
    - Generación y registro de alertas.

[22] Docker Inc. (n.d.). *What is a Container?* Docker Docs. Recuperado de https://docs.docker.com/get-started/overview/

- API REST para interacción con la interfaz de gestión y consulta de alertas.

10.1.2. Contenedor de Base de Datos

- **Motor:** PostgreSQL
- **Funcionalidad:** Almacena los datos estructurados (usuarios, salas, configuraciones) y series temporales de mediciones ambientales. También guarda el historial de alertas y umbrales definidos.
- **Configuraciones Técnicas**
  - Uso de volúmenes persistentes para evitar pérdida de datos antes reinicios.
  - Variables de entorno para credenciales y configuración de acceso.
  - Puerto estándar: 5432.
  - Se considera una instancia local para el nodo fog y una remota en la nube (Google Cloud SQL).

10.1.3. Contenedor de Frontend

- **Framework:** Vue.js
- **Funcionalidad:** Permite a los usuarios visualizar las alertas generadas, configurar umbrales y consultar datos por sala. Complementa la visualización de Grafana con herramientas de administración e interacción personalizadas.
- **Componentes clave**
  - Gestión de alertas
  - Interfaz intuitiva y simple
  - Comunicación con backend mediante API REST

10.1.4. Contenedor de Visualización

- **Herramienta:** Grafana.
- **Funcionalidad:** Permite la visualización en tiempo real de las variables ambientales, la consulta histórica por sala, y la observación de tendencias por día, semana o mes.
- **Características**
  - Integración directa con PostgreSQL

- Dashboards configurables por usuario y sala.

10.1.5. Contenedor de Simulación Sensores

- **Propósito:** Simular el comportamiento de los sensores físicos en un entorno de prueba. Está diseñado especialmente para facilitar el desarrollo, validación y demostración del sistema en las primeras etapas, sin la necesidad de contar con el hardware real (sensores y microcontroladores ESP32).
- **Funcionalidad:** Este contenedor será el encargado de simular la recolección de datos ambientales como temperatura, humedad, CO_2, ruido e iluminación, enviando esta información al backend a través de MQTT para su procesamiento, visualización y análisis en tiempo real.
- **Componentes y tareas principales:**
  - **Simulación de datos ambientales:** Genera datos ficticios que imitan las lecturas de los sensores físicos para pruebas del sistema.
  - **Envío mediante MQTT:** Publica los datos simulados al backend siguiendo el mismo procedimiento que usarán los sensores reales.
  - **Validación de alertas:** Permite verificar el correcto funcionamiento del sistema de alertas bajo diferentes condiciones simuladas.

10.2. Ventajas del enfoque basado en contenedores

El uso de contenedores en este sistema permite una arquitectura modular, flexible y fácil de mantener. A continuación, se detallan las principales ventajas de este enfoque:

- **Portabilidad:** El sistema puede replicarse fácilmente en distintos establecimientos
- **Aislamiento:** Cada componente corre en su entorno independiente, reduciendo conflictos de dependencias
- **Escalabilidad:** Permite agregar más nodos o servicios (como frontend adicional o más bases de datos locales) sin rediseñar toda la arquitectura
- **Automatización:** Compatible con procesos de CI/CD para futuras versiones del sistema.

11. Alcances y limitaciones del proyecto

Este proyecto apunta a desarrollar una primera versión funcional del sistema, enfocada en cubrir los elementos esenciales del sistema dentro de los recursos y tiempos disponibles. Si bien queda abierta la posibilidad de incorporar mejoras y de expandir sus funcionalidades en futuras versiones, el foco de esta etapa es establecer una base técnica operativa y coherente, adecuada para realizar pruebas en contextos reales y así validar su funcionamiento inicial.

11.1. Alcances

- **Monitoreo ambiental en tiempo real:** El sistema se encargará de la recolección y visualización de las variables importantes como temperatura, humedad, niveles de CO_2, ruido e iluminación, utilizando sensores físicos conectados a microcontroladores ESP32 que estarán gestionados por una red de servidores locales y la nube.

- **Almacenamiento y visualización de datos:** El sistema permitirá almacenar los datos tanto localmente (en el servidor fog) como en la nube (Google Cloud SQL), asegurando redundancia y disponibilidad histórica. Para su análisis y consulta, los datos estarán disponibles a través de dashboards interactivos en Grafana, así como mediante una interfaz web desarrollada en Vue.js, que permitirá a los usuarios visualizar variables en tiempo real, acceder a reportes históricos y recibir alertas personalizadas según su rol.

- **Alertas automáticas:** Cuando los valores de algunas de las variables monitoreadas (como temperatura o CO_2) se salgan de los rangos previamente definidos por los administradores, el sistema generará alertas automáticamente. Estas alertas ayudarán a que se tomen medidas correctivas rápidamente dentro de la sala, cómo ajustar la ventilación o la iluminación.

- **Configuración de umbrales por salas:** Cada establecimiento podrá definir umbrales personalizados para cada variable y sala, permitiendo ajustar el comportamiento del sistema según las necesidades para mejorar el entorno educativo.

- **Despliegue en contenedores Docker:** El sistema backend será entregado en contenedores Docker para facilitar la portabilidad, replicación y pruebas en distintos entornos sin importar el sistema operativo.

- **Operación offline del nodo fog:** En caso de que el sistema pierda conectividad con la nube, el sistema seguirá operando de manera autónoma desde el servidor local, guardando los datos temporalmente hasta que se recupere la conexión.

11.2. Limitaciones

- **Cobertura parcial de sensores y salas:** La implementación se limitará a un conjunto acotado de salas por establecimiento, en este caso de 5 a 10, por lo que no se contempla una cobertura masiva o simultánea en todos los espacios del recinto. Se trata de una etapa piloto para validar el sistema.

- **Interfaz de usuario básica:** Si bien Grafana permite visualizar los datos en tiempo real, la interfaz no estará personalizada a nivel avanzado ni será especialmente amigable para usuarios no técnicos. El enfoque estará en que funcione correctamente, aunque sea simple.

- **Capacitación y adopción restringida:** No se considera un plan completo de formación para todos los usuarios finales. La adopción del sistema podría requerir apoyo técnico adicional en instituciones con menor alfabetización digital.
- **Alcance técnico de mantenimiento:** Las rutinas de mantenimiento predictivo están contempladas en diseño, pero su implementación dependerá de la experiencia del personal técnico y del tiempo disponible para la operación.
- **Dependencia de conectividad institucional:** Aunque el sistema opera localmente en caso de corte de internet, su sincronización en la nube requiere una conexión estable. Instituciones con una infraestructura de red deficiente podrían enfrentar restricciones.
- **No se considera feedback de usuarios reales:** La validación de requisitos funcionales y experiencia de uso no considera aún procesos formales de recolección de feedback con actores reales (docentes, administrativos, estudiantes), por lo que algunas funcionalidades podrían no ajustarse completamente a las necesidades del entorno escolar.

12. Políticas de replicación

Para este proyecto, se establece una política de replicación orientada a garantizar la disponibilidad, integridad y continuidad operativa de los datos recolectados por los sensores ambientales desplegados en cada aula.

Tal como se expuso previamente en la descripción de la arquitectura del sistema, este se fundamenta en un modelo distribuido de tres capas: Edge–Fog–Cloud. Sobre esta base, se adopta una estrategia de **replicación pasiva y asincrónica**, con el fin de mantener copias persistentes de los datos en diferentes niveles del sistema, minimizando el riesgo de pérdida de información y garantizando su disponibilidad incluso ante fallos parciales.

Específicamente, los datos recolectados se almacenan de manera local en el nodo Fog, mediante una base de datos **PostgreSQL**, y se sincronizan periódicamente con la capa **Cloud**, que utiliza **Google Cloud SQL** como sistema gestor. Esta sincronización diferida permite que el sistema opere de forma autónoma en contextos de conectividad inestable, preservando localmente los datos hasta que la conexión se restablezca. Una vez restablecida la conexión, los datos que quedaron almacenados localmente son sincronizados automáticamente con la nube, lo que permite conservar un registro completo y confiable de las condiciones ambientales registradas durante el periodo de desconexión.

El mecanismo de replicación se implementará mediante scripts automatizados, programados para ejecutar la sincronización en intervalos definidos. Estos scripts están diseñados para detectar el estado de la red y condicionar la transferencia en función de la disponibilidad de conectividad. Esta estrategia garantiza una operación tolerante a fallos de red, previniendo la pérdida de datos críticos y asegurando la continuidad funcional del sistema.

Adicionalmente, en zonas identificadas con alta criticidad como por ejemplo, en salas con alta densidad de ocupación o con poca ventilación, se contempla una replicación física mediante sensores redundantes. Esta estrategia consiste en instalar sensores adicionales en puntos cercanos o estratégicos dentro del mismo espacio físico, con el fin de generar lecturas paralelas que permitan verificar la consistencia de los datos en tiempo real, facilitando la detección temprana de fallos, la identificación de comportamientos anómalos en sensores individuales, y la mejora de la precisión en la interpretación de las condiciones ambientales del aula.

En cuanto al modelo de consistencia de los datos, se opta por un enfoque de consistencia eventual. Esto significa que puede haber pequeñas diferencias temporales entre la información almacenada localmente (en el nodo Fog) y la que está en la nube, pero con el tiempo ambas copias se sincronizan y quedan iguales. Esta decisión permite que el sistema sea más escalable, flexible y resistente, especialmente útil en entornos donde hay conectividad inestable o recursos limitados, como puede ocurrir en muchas instituciones educativas.

Por lo tanto, la política de replicación aplicada logra un equilibrio adecuado entre eficiencia, disponibilidad de los datos y capacidad para resistir fallos. Esto se alinea con las buenas prácticas en el diseño de sistemas distribuidos actuales y asegura que el sistema pueda funcionar de manera confiable en contextos educativos reales, incluso si las condiciones de conectividad o recursos varían entre una sala y otra.

13. Políticas de tolerancia a fallos

El sistema distribuido ha sido diseñado considerando distintos escenarios de fallo que pueden afectar tanto la recopilación de datos como su procesamiento, visualización y sincronización. Esta sección describe los principales tipos de fallos previstos y las estrategias implementadas para mantener la operatividad, integridad y confiabilidad del sistema.

13.1. Fallo de sensores

Escenario: Un sensor ambiental (CO_2, temperatura, humedad, etc.) deja de transmitir datos por causas como desconexión, fallo físico, o interferencia.

Impacto:

- Pérdida de datos en tiempo real en una sala específica.
- Alertas no disparadas o decisiones incompletas por parte del sistema.

Política de tolerancia:

- **Detección automática de inactividad:** el sistema local evalúa si un sensor no ha enviado datos en un tiempo límite.
- **Alertas por sensor inactivo:** el backend genera una alerta específica para informar la pérdida del sensor.

- **Redundancia:** en salas críticas, se pueden instalar sensores duplicados para variables claves.

Implementación:

- El backend en la Raspberry Pi mantiene un registro del último timestamp de cada sensor.
- Si se detecta inactividad, se genera una alerta y se marca el sensor como inactivo en la base de datos.

13.2. Fallo de red mallada (ESP-MESH)

Escenario: Uno o varios nodos ESP32 pierden conectividad dentro de la red mesh.

Impacto:

- Los sensores afectados no logran enviar datos al broker MQTT.
- Posible aislamiento de nodos con cobertura limitada.

Política de tolerancia:

- **Reorganización automática:** el protocolo ESP-MESH permite que los nodos reorganicen la red y busquen nuevas rutas hacia el nodo raíz.
- **Reconexión programada:** los nodos intentan reconectarse automáticamente si pierden enlace.
- **Registro de conectividad:** el servidor local detecta y notifica pérdida de nodos en la red.

Implementación:

- El ESP32 está configurado para enviar mensajes de “heartbeat”.
- Si un nodo desaparece, el servidor local registra la pérdida e informa al administrador mediante alerta.

13.3. Fallo del servidor local (Fog Node)

Escenario: El servidor local (Raspberry Pi) deja de funcionar por falla de hardware, corte eléctrico o error crítico del sistema operativo**.**

Impacto:

- Interrupción total de la recopilación y procesamiento local de datos.
- Pérdida de datos temporales no sincronizados con la nube.

Política de tolerancia:

- **Arranque automático:** el sistema está configurado con políticas de reinicio (ej. systemd o docker restart).
- **Almacenamiento persistente local:** los datos se guardan en disco local antes de sincronizar.
- **Notificación remota:** si la desconexión se detecta en la nube (por falta de sincronización), se emite alerta al equipo técnico.

Implementación:

- El sistema cuenta con una rutina de autoverificación al encender y reporta su estado si recupera la conectividad.
- Datos no sincronizados se conservan en PostgreSQL local hasta restauración.

13.4. Fallo de conexión con la nube

Escenario: El servidor local no puede establecer conexión con la instancia de PostgreSQL en la nube debido a caídas de red, configuración DNS, o caídas de servicio en la nube.

Impacto:

- Imposibilidad de sincronizar datos históricos y estructurados.
- Dashboards en Grafana no se actualizan con nuevos datos.

Política de tolerancia:

- **Almacenamiento local temporal:** los datos se mantienen en PostgreSQL local hasta que se restablezca la conexión.
- **Reintentos automáticos:** scripts programados con cron ejecutan reintentos periódicos.
- **Prioridad de sincronización:** los datos se sincronizan en orden cronológico una vez que se restablece la conexión.

Implementación:

- El backend lleva registro del último envío exitoso.
- Una vez reconectado, el sistema sincroniza únicamente los datos pendientes.

13.5. Fallo en la visualización (Grafana)

Escenario: La plataforma de visualización en Grafana no está disponible temporalmente, ya sea por fallo en el servidor, error de conexión con PostgreSQL o problemas de configuración.

Impacto:

- Los usuarios no pueden acceder a los dashboards.
- Monitoreo visual en tiempo real queda suspendido.

Política de tolerancia:

- **Datos almacenados en PostgreSQL:** no se pierden datos, ya que la base es persistente.
- **Backup y recuperación:** la configuración de dashboards puede restaurarse desde archivos de backup o exportaciones previas.

Implementación:

- Mediante Docker, Grafana se reinicia automáticamente ante fallos.

13.6. Fallo del contenedor Docker

Escenario: Algún contenedor crítico (backend, frontend, base de datos, simulador de sensores o Grafana) falla por error interno, mal deployment, actualización fallida o conflictos de dependencia.

Impacto:

- Pérdida de servicio en el componente afectado.
- Interrupción parcial del sistema, especialmente si el contenedor de backend o base de datos falla.

Política de tolerancia:

- Uso de políticas restart: always o on-failure en docker-compose.yml.
- Backups regulares de configuración de Grafana, volúmenes de PostgreSQL y exportación de dashboards.

Implementación:

- Configurar contenedores con volúmenes persistentes.
- Implementar notificaciones por caída de servicios.
- Habilitar reintentos automáticos y logging por contenedor.

13.7. Fallo de sincronización programada (cron)

Escenario: El script de sincronización entre el fog node y la nube no se ejecuta correctamente debido a errores de red, permisos, o fallos en la configuración del cronjob.

Impacto:

- Acumulación de datos no sincronizados.
- Retraso en la visualización histórica en la nube y dashboards.

Política de tolerancia:

- Verificación del estado de sincronización mediante logs locales.
- Script de validación que asegure que cada lote de datos fue efectivamente enviado.
- Notificación al administrador si no se sincroniza en un intervalo definido.

Implementación:

- Crear archivos de lock y logs de sincronización.
- Reintento automático ante errores HTTP o fallos en el envío.
- Integración con sistema de alertas si la sincronización falla más de N veces consecutivas.

14. Costos del proyecto

En el presente apartado se detallan todos los costos relacionados al proyecto, tanto los costos de implementación que son los gastos necesarios para la puesta en funcionamiento del sistema, incluyendo componentes, adaptadores y proyecciones asociadas de cara a la misma implementación, como los costos de desarrollo que son los gastos operacionales y de mantención de los dispositivos, software externo, así como la mano de obra necesaria para el funcionamiento del sistema.

14.1. Costos de implementación

14.1.1. Presupuesto aproximado de recursos en la nube

14.1.1.1. Servicio de computación

Recurso	Detalle técnico	Cantidad	Costo estimado mensual
Servidor backend en la nube	Servidor virtual e2-standard-4 (4 vCPUs, 16 GB de RAM)	1 servidor	$87.400 - $123.700

Instancia de desarrollo	VM liviana para pruebas y mantenimiento (e2-small)	1 nodo	$12.000 - $20.000
Subtotal computación			**$99.400 - $143.700**

Tabla 1: Tabla de costos de servicios de computación[23].

14.1.1.2. Almacenamiento y base de datos

Recurso	Detalle técnico	Cantidad	Costo estimado mensual
Google Cloud SQL (PostgreSQL)	Instancia gestionada con 30 GB de almacenamiento SSD	1 instancia	$10.800 - $12.000
Backups automáticos	5 GB diarios con retención de 30 días	Incluido	$10.000 - $15.000
Google Cloud Storage	100 GB para logs, exportaciones o archivos extra	1 bucket	$2.000 -$3.000
Subtotal almacenamiento			**$22.800 - $30.000**

Tabla 2: Tabla de costos de almacenamiento y base de datos.

14.1.1.3. Transferencia de datos y red

Recurso	Detalle técnico	Costo estimado mensual
Transferencia de datos	200GB/mes aprox. entre nodos y la nube	$20.000 - $35.000

[23] Google Cloud. (s. f.). *Precios de Compute Engine*. https://cloud.google.com/compute/all-pricing

Balanceo de carga	Reglas básicas de HTTPS y routing	$10.000 - $16.000
Subtotal transferencia y red		**$30.000 - $51.000**

Tabla 3: Tabla de costos de servicios de transferencia de datos y red[24].

14.1.1.4. Servicios de seguridad

Recurso	**Detalle técnico**	**Costo estimado mensual**
Certificado SSL/TLS	Uso de certificados gestionados en GCP	Incluido sin costo
Firewall y reglas de acceso	Reglas IP y roles restringidos	Incluido sin costo
Autenticación por roles	IAM de GCP, niveles de usuario en sistema	Incluido sin costo
Subtotal seguridad		**$0**

Tabla 4: Tabla de costos de servicios de seguridad.

14.1.1.5. Monitoreo y mantenimiento

Recurso	**Detalle técnico**	**Costo estimado mensual**
Stackdriver/Cloud Monitoring	Alertas y métricas básicas del sistema	$0 - $5.000
Mantenimiento técnico programado	Gestión y revisión mensual	$15.000 - $25.000

[24] Google Cloud. (s. f.). *Precios de red de VPC*. https://cloud.google.com/vpc/network-pricing?hl=es_419#egress

Subtotal monitoreo y mantenimiento		**$15.000 - $30.000**

Tabla 5: Tabla de costos de servicios de monitoreo y mantenimiento[25].

14.1.1.6. Resumen de costos mensuales

Categoría	Costo mínimo	Costo máximo
Computación	$99.400	$143.700
Almacenamiento y BD	$22.800	$30.000
Transferencia y red	$30.000	$51.000
Seguridad	$0	$0
Monitoreo y mantenimiento	$15.000	$30.000
Total estimado	**$167.200**	**$254.700**

Tabla 6: Tabla resumen de costos mensuales de los servicios de la nube.

14.1.2. Instalación y configuración

Actividad	Perfil responsable	Costo por hora	HH	Costo Total
Configuración de software back-end	Ingeniero DevOps	$16.000-$19.000	8	$128.000-$152.000

[25] Google Cloud. (s. f.). *Precios de Cloud Monitoring*. https://cloud.google.com/stackdriver/pricing?hl=es-419

Instalación de base de datos	Administrador de Base de Datos	$9.500-$15.500	4	$38.000-$62.000
Configuración de entorno front-end	Desarrollador Front-End	$13.500-$16.500	4	$54.000-$66.000
Instalación de sistemas necesarios	Administrador de sistemas	$4.500-$8.500	8	$36.000-$68.000
Verificación y pruebas iniciales	QA	$7.500-$11.000	8	$60.000-$88.000
Total estimado				**$316.000-$436.000**

Tabla 7: Estimación de costos por instalación y configuración del sistema.

Los cálculos de costo se realizaron considerando el sueldo mínimo y máximo de cada responsable relacionado a la actividad en Chile.

14.1.3. Capacitación inicial

Este punto considera tanto manuales como el tiempo invertido en entrenar a los usuarios finales y al personal técnico.

Actividad	Perfil responsable	Costo por hora	HH	Costo Total
Elaboración de manuales de uso de la aplicación	Equipo de desarrollo	$10.000	10	$100.000
Capacitación del docente y al personal técnico	Equipo de desarrollo	$10.000	20	$200.000
Total Estimado				**$300.000**

Tabla 8: Estimación de Costos por Capacitación Inicial

14.1.4. Componentes y unidades

A continuación se presentan el coste unitario y de mantenimiento de los componentes que constituirán el sensor, así como todas las unidades de alimentación y comunicación necesarias

Componente	Descripción	Costo Unitario (CLP)	Mantenimiento Unitario (CLP)
ESP32	Microcontrolador con conectividad Wifi y Bluetooth.	$7.356 - $11.990	$1.000 - $1.500
DHT22	Sensor digital de temperatura y humedad de alta precisión.	$4.000 - $6.100	$1.000 - $1.300
MH-Z19	Sensor de infrarrojos no dispersivo (NDIR) que mide la concentración de dióxido de carbono (CO2).	$25.421 - $30.767	$2.500 - $3.000
KY-038	Sensor con micrófono para la detección de sonidos.	$1.990 - $3.675	$500 - $800
LDR	Resistor dependiente de la luz, cuya resistencia cambia en respuesta a la luz incidente	$1.990 - $8.549	$500 - $1.000
Adaptador de corriente 5v	Dispositivo que se utiliza para convertir la corriente eléctrica en fuente de energía a una tensión de salida de 5 voltios.	$3.412 - $5.990	$500 - $800
Baterías 18650	Baterías recargables de iones de litio, de alta densidad energética y durabilidad.	$2.827 - $4.990	$300 - $500
Módulo TP4056 con protección	Controlador de carga de baterías de iones de litio. Ofrece protección contra sobrecarga y sobre-descarga, indicador LED de carga y corriente de carga ajustable.	$2.290 - $6.990	$300 - $600

Conmutador automático	Dispositivo que permite la conmutación automática entre diferentes fuentes de energía, que garantiza un suministro continuo y confiable en caso de fallos en la fuente principal.	$11.396 - $15.128	$800 - $1.000
Raspberry Pi 4	Servidor local Fog Node	$79.990 - $100.690	$12.000 - $15.000

Tabla 9: Tabla de costos de implementación.

Una vez observados los costos de la tabla anterior de componentes, se puede estimar que los costos totales durante un ciclo de funcionamiento regular de aproximadamente 1 año (que incluye tanto la implementación inicial como los costos mensuales acumulados por mantenimiento) son los siguientes:

Sensor completo (por cada sala de clases): Este conjunto incluye ESP32, DHT22, MH-Z19, KY-038, LDR, adaptador de corriente, baterías, TP4056 y conmutador automático.

- **Costo unitario mínimo (CLP):** $60.682
- **Costo unitario máximo (CLP):** $94.179
- **Mantenimiento mensual mínimo (CLP):** $7.400
- **Mantenimiento mensual máximo (CLP):** $10.500
- **Costo total anual mínimo:** $60.682 + ($7.400 x 12) = $149.482
- **Costo total anual máximo:** $94.179 + ($10.500 x 12) = $220.179

Servidor Local Fog (Raspberry Pi 4): Un nodo fog por cada establecimiento.

- **Costo unitario mínimo (CLP):** $79.990
- **Costo unitario máximo (CLP):** $100.690
- **Mantenimiento mensual mínimo (CLP):** $12.000
- **Mantenimiento mensual máximo (CLP):** $15.000
- **Costo total anual mínimo:** $79.990 + ($12.000 x 12) = $223.990
- **Costo total anual máximo:** $100.690 + ($15.000 x 12) = $280.690

14.2. Proyección de costos para implementación a escala

Si el sistema se implementa por ejemplo en 10 salas de clases de un mismo establecimiento requerirá:

- **10 sensores completos**, con un costo total entre $1.494.820 y $2.201.790
- **1 servidor local,** con un costo total entre $223.990 y $280.690
- **2 días de equipo de instalación (5 salas por día),** con un costo total entre $632.000-$872.000
- **Manuales de uso y capacitación**, con un costo total de $300.000

Total estimado anual (10 salas + 1 servidor):

- **Costo mínimo total:** $1.494.820 + $223.990 + $632.000 + $300.000 = $2.650.810
- **Costo máximo total:** $2.201.790 + $280.690 + $672.000 + $300.000= $3.454.480

14.3. Costos de desarrollo

A continuación se presenta una estimación de las horas-hombre (HH) requeridas para implementar, operar y mantener el sistema durante un año, asumiendo un costo promedio de **$10.000 CLP por hora** (lo realizará el equipo de trabajo).

Categoría	Tarea	HH Estimadas	Frecuencia	Total HH Anual	Costo (CLP)
Diseño y desarrollo	Análisis de requerimientos y planificación técnica	60 HH	Única (fase inicial)	60 HH	$600.000
	Desarrollo de backend en Go	100 HH	Única	100 HH	$1.000.000
	Programación de sensores y comunicación MQTT	80 HH	Única	80 HH	$800.000
	Configuración de Grafana	60 HH	Única	60 HH	$600.000
	Configuración de	40 HH	Única	40 HH	$400.000

	PostgreSQL				
Operación y mantenimiento	Monitoreo del sistema y supervisión de alertas	8 HH/mes	Mensual	96 HH	$960.000
	Calibración de sensores (ruido, CO2, luz, temperatura)	10 HH/mes	Mensual	120 HH	$1.200.000
	Actualización remota de firmware (ESP32)	10 HH/trimestre	Trimestral	40 HH	$400.000
Mejoras y soporte	Mantenimiento correctivo y mejoras en dashboards	8 HH/trimestre	Trimestral	32 HH	$320.000

Tabla 10: Tabla de costos Horas - Hombre.

Categoría	**Total HH Anual**	**Costo total (CLP)**
Diseño y desarrollo	340 HH	$3.400.000
Operación y mantenimiento	256 HH	$2.560.000
Mejoras y soporte	32 HH	$320.000
Total General	**628 HH**	**$6.280.000**

Tabla 11: Tabla de costos horas totales Horas - Hombre anual por categoría.

14.4. Costos marginales

En el proyecto existen ciertos costos marginales asociados, los cuales se detallan a continuación:

- **Google Cloud SQL:** El costo mensual se incrementa aproximadamente en $1.000 CLP por cada GB adicional de almacenamiento SSD por sobre los 30 GB.
- **Google Cloud Storage:** El costo mensual incrementa aproximadamente en $300 CLP por cada 10 GB adicionales.
- **Mantenimiento técnico adicional:** Se debe considerar un monto para imprevistos en caso de reposición de alguno de los dispositivos ante eventual fallo.

15. Conclusión

El presente informe ha desarrollado una propuesta integral para la implementación de un sistema distribuido de monitoreo ambiental en aulas académicas, respondiendo a la necesidad urgente de mejorar las condiciones físicas en los espacios educativos. A través de una arquitectura basada en el modelo Edge–Fog–Cloud, el sistema propuesto permite recolectar, procesar, visualizar y analizar datos en tiempo real sobre variables críticas como temperatura, humedad, concentración de CO_2, ruido e iluminación, contribuyendo directamente al bienestar de estudiantes y docentes.

La solución fue diseñada para ser escalable, resiliente y adaptable a diferentes contextos escolares, especialmente considerando ambientes con infraestructura limitada. Para ello, se utilizaron tecnologías abiertas y eficientes como sensores conectados a microcontroladores ESP32, redes malladas (ESP-MESH), el protocolo MQTT, bases de datos PostgreSQL, visualización con Grafana y una interfaz de gestión desarrollada en Vue.js, todo orquestado mediante contenedores Docker para facilitar su despliegue y mantenimiento.

Asimismo, se detallaron los aspectos fundamentales de diseño, implementación y operación del sistema, incluyendo políticas de tolerancia a fallos, estimaciones de costos, herramientas utilizadas y planificación de recursos humanos. Se diseñó una base de datos robusta, se definieron mecanismos de sincronización entre capas, y se establecieron umbrales configurables para la emisión automatizada de alertas ambientales. También se consideraron criterios de seguridad, privacidad y sostenibilidad a largo plazo.

En suma, esta propuesta representa una primera versión funcional sólida y coherente, preparada para ser validada en contextos reales. El sistema constituye una herramienta efectiva para promover entornos escolares más confortables, inclusivos y saludables, con un alto potencial de impacto tanto en el rendimiento académico como en la calidad de vida dentro del aula. Su modularidad y capacidad de extensión garantizan que podrá adaptarse a futuras necesidades y desafíos del entorno educativo.

16. Preguntas para el cliente y supuestos

Con el objetivo de asegurar que el sistema de monitoreo ambiental cumpla con las necesidades reales de los establecimientos educativos, se plantea el siguiente conjunto de preguntas dirigidas al cliente:

- **¿Cuántas salas de clases desea monitorear inicialmente?**

  Se estima un despliegue inicial en 5 a 10 salas por establecimiento educativo.

- **¿Qué variables ambientales se desean monitorear en cada sala?**

  Se considerarán temperatura, humedad, niveles de CO_2, ruido ambiental e iluminación.

- **¿Con qué frecuencia es necesario medir y actualizar los datos de los sensores?**

Se configurará una frecuencia de medición cada 15 minutos, ajustable según las necesidades del establecimiento.

- **¿Existen valores críticos conocidos para las variables monitoreadas (ej. umbrales de CO_2 o temperatura)?**

  Se asumirán umbrales iniciales basados en estándares de confort ambiental. Por ejemplo: CO_2 > 1500 ppm es crítico; T° fuera del rango 18-28°C es desfavorable.

- **¿El establecimiento cuenta con acceso a internet estable en las salas?**

  Se asume que hay conectividad WiFi básica y estable, suficiente para la operación de una red ESP-MESH y sincronización periódica con la nube.

- **¿Qué dispositivos utilizarán los usuarios para acceder a la plataforma de visualización?**

  Se utilizarán computadores de escritorio, notebooks, teléfonos o tablets conectados a internet vía navegador web. No se considera app móvil en esta etapa.

- **¿Quién será el responsable del monitoreo de alertas y mantenimiento del sistema dentro del colegio?**

  Se estima que el personal administrativo y técnico del establecimiento será capacitado para realizar el monitoreo y el mantenimiento básico mensual.

- **¿Existe interés en incluir variables adicionales en el futuro (como presencia de polvo o luminosidad exterior)?**

  Se asume que existe interés futuro, por lo que el sistema será modular y permitirá la integración de nuevos sensores compatibles con ESP32.

- **¿El sistema debe integrarse con plataformas existentes del establecimiento (por ejemplo, sistemas académicos)?**

  En esta etapa no se contempla integración con otros sistemas. Se prevé como posible requerimiento en futuras versiones.

- **¿Qué nivel de criticidad desea aplicar para cada variable ambiental?**

  Se adoptará un esquema de dos niveles: preventivo y crítico, con visualización diferenciada por colores y alertas.

- **¿Se desea contar con respaldo energético en caso de corte eléctrico?**

  Se incluirán baterías de respaldo con una autonomía mínima de 2 horas por sensor.

- **¿Se requiere visualizar datos históricos además del monitoreo en tiempo real?**

  Sí. Se habilitarán gráficos históricos por hora, día y semana en dashboards de Grafana.

- **¿Cuál es el periodo mínimo por el que deben almacenarse los datos?**

  Se asumirá una retención de datos en la nube por al menos 2 años.

- **¿Qué medidas de seguridad y privacidad son necesarias para los datos?**

  Se cifrarán las comunicaciones con TLS y se implementará autenticación por roles para limitar el acceso a los dashboards.

- **¿Desean que las alertas sean sonoras, visuales o ambas?**

  Se configurarán como visuales por defecto, con posibilidad de configurar alertas sonoras o silenciosas según cada sala.