UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

“Proyecto de monitoreo y regulación de la humedad ambiental para evitar la formación de moho y salitre en paredes”

Equipo: **Hilda Albarracín**
Mayling Álvarez
Antonella Butrón
Ana Gutiérrez
Asignatura: Proyecto II

Profesor: Diego Alberto Aracena Pizarro

ARICA, 25 de Noviembre 2025

Historial de Cambios

Fecha	Versión	Descripción	Autor(es)
07/10/2025	1.0	Formato	Ana Gutiérrez
14/10/2025	1.1	Panorama General	Mayling Álvarez Ana Gutiérrez
16/10/2025	1.2	Organización del Proyecto	Hilda Albarracín Mayling Álvarez Antonella Butrón Ana Gutiérrez
27/10/2025	1.3	Planificación de los Procesos de Gestión	Antonella Butrón Hilda Albarracín
28/10/2025	1.4	Conclusión y referencias	Mayling Álvarez Ana Gutiérrez Antonella Butrón Hilda Albarracin
04/11/2025	1.5	Planificación de procesos Técnicos	Antonella Butrón Ana Gutiérrez
24/11/2025	1.6	Planificación de procesos de soporte, Soluciones propuestas, trabajo a futuro	Mayling Álvarez Ana Gutiérrez Antonella Butrón Hilda Albarracin

Índice

Índice de tablas

Índice de figuras

Proyecto II

1. Panorama General

1.1. Resumen del Proyecto

1.1.1. Problemática

La humedad en el ambiente es un problema que puede afectar tanto en la integridad de estructuras como en la salud de las personas que las habitan. Esta humedad puede absorberse por paredes, techos o piso, acumulando agua, lo cual favorece la formación de moho y sales, que provocan el deterioro de los materiales, además de posibles enfermedades respiratorias.

Este es un problema vigente en la ciudad de Arica. Aunque su principal característica es el clima desértico, con temperaturas relativamente cálidas (entre 15°C y 26°C), además de precipitaciones casi nulas, la humedad se presenta gracias al detalle de que es costera, cual incluye nubes y neblinas matinales, con presencia de sales, especialmente en las zonas más costeras con “camanchacas”.

Este clima es perfecto para la formación de los efectos que se quiere evitar - el moho y la sal - el clima cálido y húmedo genera buenas condiciones para la formación de moho, y la humedad entrante, como es costera, suele venir con sales, que luego se acoplan a las paredes y al evaporarse el agua dentro de estas deja la sal incrustada en las estructuras. Estas condiciones también pueden agravarse para construcciones antiguas o con escasa protección térmica.

Este es un problema con poca visibilidad sobre todo porque los efectos estructurales del moho y la sal no son tan visibles, incluso suele aparecer en zonas ocultas del hogar. Esta falta de presencia provoca que no sea tratado con frecuencia, provocando daños en el bienestar del hogar y sus habitantes, que a largo plazo serán más difíciles de tratar.

1.1.2. Propósito

Este proyecto busca mitigar los efectos negativos de la humedad en los hogares de la región de Arica, principalmente para proteger la integridad de las estructuras, y de paso la salud de los habitantes. Por ello, se pretende medir y controlar la humedad en ambientes cerrados, además de determinar si la formación de moho y sal es probable, en techos, pisos y paredes, con tal de generar conciencia sobre el mantenimiento preventivo en viviendas. De esta manera aportando en evitar las consecuencias de la humedad en los hogares, contribuyendo en la seguridad estructural y salud ambiental.

1.1.3. Alcance

- **Aplicación:** espacios interiores como viviendas, oficinas, aulas o bodegas.
- **Distancia de transmisión:** hasta donde alcance el wifi del usuario.
- **Precisión:** margen de error de aproximadamente **±3%** en la medición de humedad relativa.
- **Tipo de uso:** preventivo y experimental (no correctivo ni industrial).

No abarca:

- Reparo de daños ya existentes.
- Intervenciones reales en edificios o obras a escala completa.
- Diagnóstico médico ni tratamiento de afecciones respiratorias.
- Producción industrial o comercial del sistema.

1.1.4. Objetivos

General

Diseñar e implementar un sistema de prevención y control de la humedad en el ambiente con el fin de minimizar sus efectos negativos (específicamente moho y sal) sobre las estructuras dentro de un área cerrada.

Específicos

- Investigar sobre la humedad y sus efectos en ambientes cerrados.
- Analizar los distintos tipos de sensores y actuadores disponibles.
- Diseñar la arquitectura del sistema de prevención y control.
- Implementar y programar el sistema utilizando Raspberry Pi.
- Probar y evaluar el funcionamiento del sistema en un entorno controlado.
- Documentar resultados y conclusiones del proyecto realizado.

1.2. Suposiciones y restricciones

Suposiciones

- El ambiente para aplicar el proyecto ha de ser un entorno cerrado siempre.
- No hay caídas de wifi.
- Los integrantes del equipo tendrán el conocimiento necesario para llevar a cabo la realización del proyecto

Restricciones

- Realizar el proyecto en el tiempo establecido
- Realizar el proyecto con los sensores y recursos proveídos por el departamento. En caso de comprarlos, estos no deben superar los $20.000.
- La maqueta debe ser realizada usando elementos reciclables y máquinas CNC existentes en FI ING 2030 o en su efecto utilizando meta quest 3 de DICI.
- Asistencia del 100% en las sesiones evaluativas del proyecto.

1.3. Entregables del Proyecto

- Maqueta virtual
- Esquema de la solución
- Manual de usuario
- Poster
- Informes de avance
- Presentaciones
- Bitacoras

2. Organización del Proyecto

2.1. Personal y entidades internas

Personal

- Hilda Albarracín
- Mayling Álvarez
- Antonella Butrón
- Ana Gutiérrez

2.2. Roles y Responsabilidades

- **Jefe de proyecto**: Representante del equipo, supervisa y organiza el progreso del proyecto.
- **Ensamblador**: Encargado del montaje y el armado de las piezas, monitorea el cumplimiento de las funcionalidades, en conjunto con el programador.
- **Programador**: Encargado del área de la codificación y funcionamiento, en colaboración del ensamblador.
- **Documentador**: Encargado de registrar el avance del proyecto, junto con la redacción de los informes.
- **Diseñador**: Encargado de la creación de modelos y diagramas del proyecto.

Tabla Roles y Responsabilidades

Tabla 1. Roles y responsabilidades.

Rol	Responsable	Involucrados
Jefe	Hilda	Mayling
Ensamblador	Antonella	Mayling
Programador	Mayling	Ana Gutiérrez
Documentador	Ana Gutiérrez	Antonella
Diseñador	Hilda	Antonella

2.3. Mecanismos de Comunicación

- **Whatsapp**: Utilizado para comunicación inmediata entre los integrantes del equipo, permitiendo la resolución de dudas rápidamente y la coordinación diaria.
- **Notion**: Para la planificación del proyecto, documentación y seguimiento del proyecto. Además de la asignación de tareas y su seguimiento.
- **Correo electrónico:** Para el envío de información formal, compartir documentos y comunicación con el docente.

3. Planificación de los Procesos de Gestión

3.1. Planificación inicial del proyecto

3.1.1. Planificación de estimaciones

Planificación de estimaciones

Tabla 2. Costos de hardware.

Hardware	Cantidad	Costo
GrovePi+ Starter Kit for Raspberry Pi	1	$243.939
ESP8266	1	$10.000
ProtoBoard	2	$6.000
Deshumidificador	1	$6.590
Sensor de conductividad	1	$3.000
Raspberry PI 4 Model B/ 8GB RAM	1	$164.990
Ventiladores de extracción	1	$9.000
Luz UV	1	$7.000
HP Pavillion Plus (Costo de Uso)	1	$120.000
MSI Katana GF66 (Costo de Uso)	1	$120.000
Acer (Costo de Uso)	1	$60.000
Lenovo Thinkpad x390 yoga (Costo de Uso)	1	$ 120.000
SD 32 GB con adaptador	1	$8.396
Total:	14	$878.915

(Activos preexistentes del equipo)

Tabla 3. Costos de software.

Software	Costo
Software de desarrollo	$0
Python	$0
Blender	$0
Unity	$0

Tabla 4. Tiempo.

	Tiempo
Diseño y construcción de la maqueta	2 semanas
Programación	4 semanas

3.1.2. Planificación de recursos humanos

Tabla 5. Planificación de recursos humanos.

Rol	**Cantidad**	**Sueldo/Hora**	**Horas Totales**	**Costo**
Jefe	2	$25.000	4.5	$225.000
Ensamblador	2	$12.000	10	$240.000
Programador	2	$18.000	12	$432.000
Documentador	3	$10.000	6	$180.000
Diseñador	2	$15.000	8	$240.000
Total:	11	-	-	$1.317.000

3.2. Lista de Actividades

3.2.1. Carta Gantt

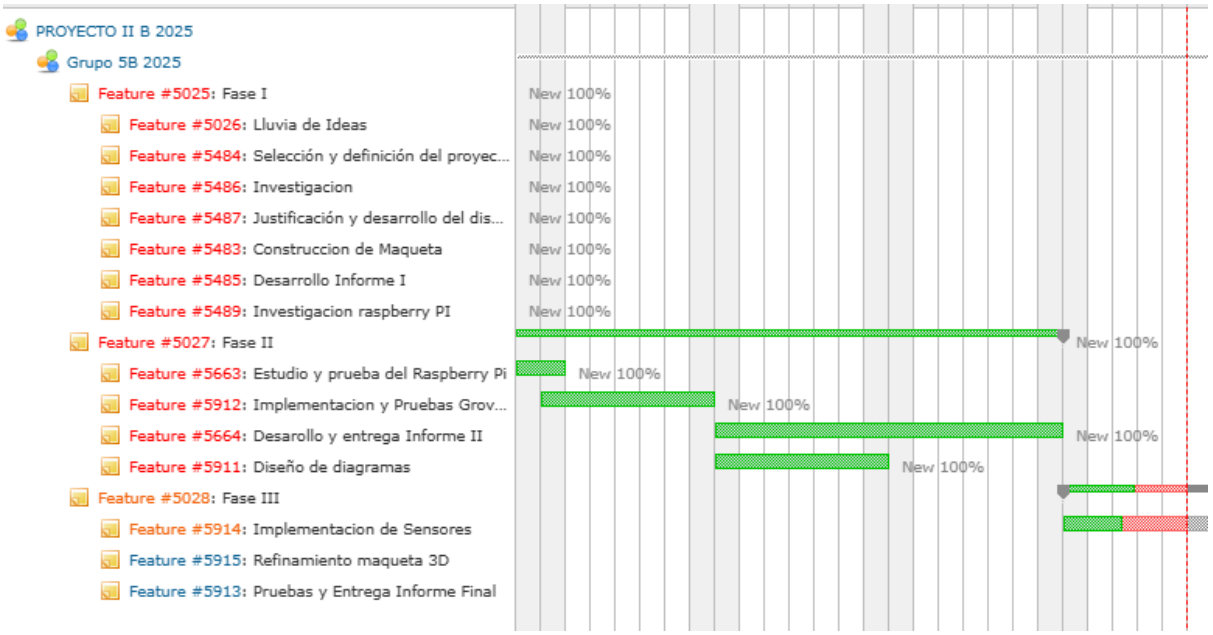


Figura 1. Carta Gantt.

3.2.2. Actividades de trabajo

Planificación del proyecto

- Lluvia de ideas
- Selección y definición del proyecto
- Investigación de la problemática, establecer panorama general, organización del proyecto, etc.

Ejecución del Proyecto

- Análisis
    - Desarrollo del informe
        - Investigar sobre las funcionalidades del Raspberry
        - Establecer objetivos y requerimientos
        - Planificación de los roles organización del proyecto
        - Justificación y desarrollo del diseño
        - Definir los recursos necesarios
        - Determinar los lenguajes de programación a utilizar
- Diseño
    - Diseño de la arquitectura del sistema
    - Construcción de la maqueta
    - Diseño gráfico de la interfaz gráfica
- Codificación
    - Establecer conexiones con los dispositivos necesarios y realizar pruebas de funcionamiento
    - Desarrollar interfaz gráfica para el usuario de la aplicación móvil

- Programar la funcionalidades de la aplicación y el sistema
- Pruebas de funcionamiento del sistema y la aplicación.

Cierre del Proyecto

- Documentación final con las conclusiones y resultados del proyecto.
- Presentación final de los resultados.
- Firma de actas de conformidad.

3.2.3. Asignación de tiempo

- Planificación del proyecto: 2 semanas.
- Ejecución del proyecto: 3 a 4 semanas.
- Cierre de proyecto: 1 semana.

3.3. Planificación de Riesgos

En la siguiente tabla se presentan los posibles riesgos del proyecto. A cada riesgo se le da un porcentaje de ocurrencia (0 - 100%), un nivel de impacto (1 - 4) y una acción remedial, como solución para el riesgo, si es que llegara a ocurrir.
Los niveles para medir el impacto son:

1. Catastrófico: Requiere resolución inmediata, debido a que puede hacer que el proyecto se detenga durante un tiempo indeterminado.
2. Crítico: Debe solucionarse de inmediato, puesto que puede retrasar el proyecto varias etapas.
3. Marginal: Es necesario resolver el riesgo a la brevedad, ya que puede causar un retraso en una etapa del proyecto.
4. Despreciable: Puede resolverse en cualquier momento.

Tabla 6. Gestión de riesgos.

Riesgos	**Probabilidad ocurrencia**	**Nivel de impacto**	**Acción remedial**
Tiempo insuficiente para el término del proyecto	20%	1	Priorizar funcionalidades fundamentales dejándolas totalmente funcionales.
Funcionamiento incorrecto de los recursos tecnológicos	40%	3	Solicitar a los ayudantes nuevo material. En caso de ser una adquisición externa, evaluar las posibilidades de cambiar el recurso.
Cambios de los requisitos	40%	2	Mitigar lo más posible el impacto del cambio del requisito en los otros.
Asignación de tareas desbalanceada	20%	3	Redistribuir las tareas y priorizar las más críticas para aliviar la carga de trabajo de los integrantes.
Incumplimiento de las tareas	40%	2	Reorganización priorizando las tareas atrasadas.
Integrante del equipo se	30%	3	Reorganizar al equipo para avanzar en las

ausenta a una sesión de trabajo			tareas del personal ausente.
Errores en el código difíciles de detectar	30%	1	Solicitar ayuda a expertos
La capacidad del equipo está por debajo de lo que pide el proyecto	20%	1	Reasignación de tareas tomando en cuenta las capacidades y competencias reales de los integrantes.
Necesidad de gastar más de lo presupuestado	60%	2	Priorizar los gastos más necesarios para cumplir los objetivos del proyecto y ayudar al funcionamiento del sistema.
Falta de información disponible para aprender a usar las herramientas	45%	1	Realizar reuniones de equipo para investigar más exhaustivamente en conjunto. En casos extremos, evaluar la posibilidad de cambio de las herramientas.
Prototipo estropeado	30%	1	Tratar de reparar el prototipo. En casos extremos, reconstruir el prototipo e informar al docente.
Funcionamiento incorrecto del sistema final	30%	1	Realizar pruebas de funcionamientos a cada módulo del sistema. Tratar de solucionar los errores de los recursos alcanzables, lo más rápido posible.

Factores de riesgo

Tabla 7. Factores de riesgo.

Tipo de riesgo	**Indicadores potenciales**
Tecnología	Fallo de la tecnología utilizada e imposibilidad de dar solución a ello.
Personas	Falta de capacidad del equipo, falta de disponibilidad para trabajar. Malas relaciones entre los integrantes. Falta de diálogo entre los integrantes para llegar a acuerdos.
Organizacional	Mala organización con respecto a tiempos y asignación de tareas.
Herramientas	Falta de concordancia entre los integrantes con respecto a qué herramientas utilizar y cuáles no utilizar.
Requerimientos	Constantes cambios en los requerimientos del proyecto.
Estimación	Fallo en el cumplimietos de los tiempos estimados y los errores encontrados.

4. Planificación de procesos Técnicos

4.1. Modelo de Proceso

4.1.1. Análisis

4.1.1.1. Especificación de Requerimientos

4.1.1.1.1. Requisitos funcionales

- **Monitoreo de la humedad, el moho y la sa**l:
    - RF01: El sistema debe medir la humedad del ambiente interno y compararlo con el umbral configurado.
    - RF02: El sistema debe mostrar los niveles de humedad y sal medidos en la habitación en la interfaz de usuario.
    - RF03: Si la humedad excede el umbral configurado, el sistema debe encender el deshumidificador para la detección de sal en el ambiente.
    - RF04: Si el deshumidificador recolecta el agua necesaria para la medición, se inicia la detección de sal en el ambiente.
    - RF05: El sistema debe incorporar umbrales de alerta por defecto para la humedad y sal, los cuales podrán ser modificados por el usuario a través de la interfaz.
    - RF06: Si la humedad deja de exceder el umbral configurado, el sistema debe apagar el deshumidificador.
- **Control de la humedad**:
    - RF07: El sistema debe identificar el nivel de sal dentro de un rango para poder activar las medidas de control.
    - RF08: El sistema debe permitir desactivar o posponer las medidas de control a través de la interfaz de usuario.
    - RF09: Si es que el usuario no desactiva o pospone las medidas de control, el sistema debe activar automáticamente los ventiladores y la luz UV.
    - RF10: El sistema debe notificar al usuario cuando se necesite que este vacíe el agua almacenada en el contenedor del humidificador.
    - RF11: El sistema debe identificar si el nivel de sal sigue siendo alto después de aplicar medidas de control, comparando el valor actual con un umbral predefinido.
    - RF12: Si los niveles de sal y humedad siguen siendo altos, el sistema debe avisar el nivel de gravedad al usuario a través de la interfaz.
    - RF13: Si los niveles de sal y humedad siguen siendo altos, el sistema debe mostrar recomendaciones ,sobre las medidas de acción a tomar, al usuario a través de la interfaz.
- **Modo de envió de datos**:

    - RF14: El sistema debe enviar los datos a la aplicación móvil mediante conexión Wi-Fi.

- **Almacenamiento de información en la base de datos**:
    - RF14: El sistema debe almacenar los datos registrados en una base de datos (local o en la nube).
- **Interfaz de usuario**:
    - RF15: La interfaz debe mostrar en una sola pantalla el estado del sistema, las mediciones de humedad y sal.
    - RF16: La interfaz debe mostrar alertas cuando se detecten altos niveles de sal, permitiendo al usuario desactivar o posponer las medidas de control.
    - RF17: La interfaz debe permitir al usuario consultar el historial de lecturas registradas.

4.1.1.1.2. Requisitos no funcionales

- **Rendimiento**:
    - RNF01: El sistema debe medir la humedad del ambiente cada 5 segundos.
    - RNF02: Las alertas o notificaciones deben mostrarse en la interfaz en menos de 2 segundos tras su detección.
- **Usabilidad**:
    - RNF03: La aplicación debe tener una interfaz clara e intuitiva.
- **Confiabilidad**:
    - RNF04: El sistema debe estar operativo la mayoría del tiempo para garantizar el funcionamiento continuo y estable.
    - RNF05: El historial de datos debe respaldarse periódicamente para evitar pérdidas.
- **Seguridad**:
    - RNF06: La comunicación entre el Raspberry Pi 4B y la aplicación debe realizarse mediante un protocolo seguro.

4.1.1.2. Modelo de Diseño

El modelo del diseño muestra cómo el sistema monitorea continuamente la humedad y la salinidad del entorno, y cómo utiliza estos datos para determinar si existe un nivel de gravedad que requiera notificar al usuario para invitarlo a seguir uno de los planes de acción sugeridos según el nivel determinado de su situación ambiental. Cada funcionalidad va representada en una burbuja como caso de uso, que van desde el monitoreo, encendido y apagado, y los avisos. El diagrama cubre el sistema en caso de encontrarse en la etapa dos del proyecto, donde se implementan los ventiladores con la luz UV, además de integrar los sensores necesarios para el monitoreo, más el dispositivo de notificación para el usuario, en este caso su teléfono celular.

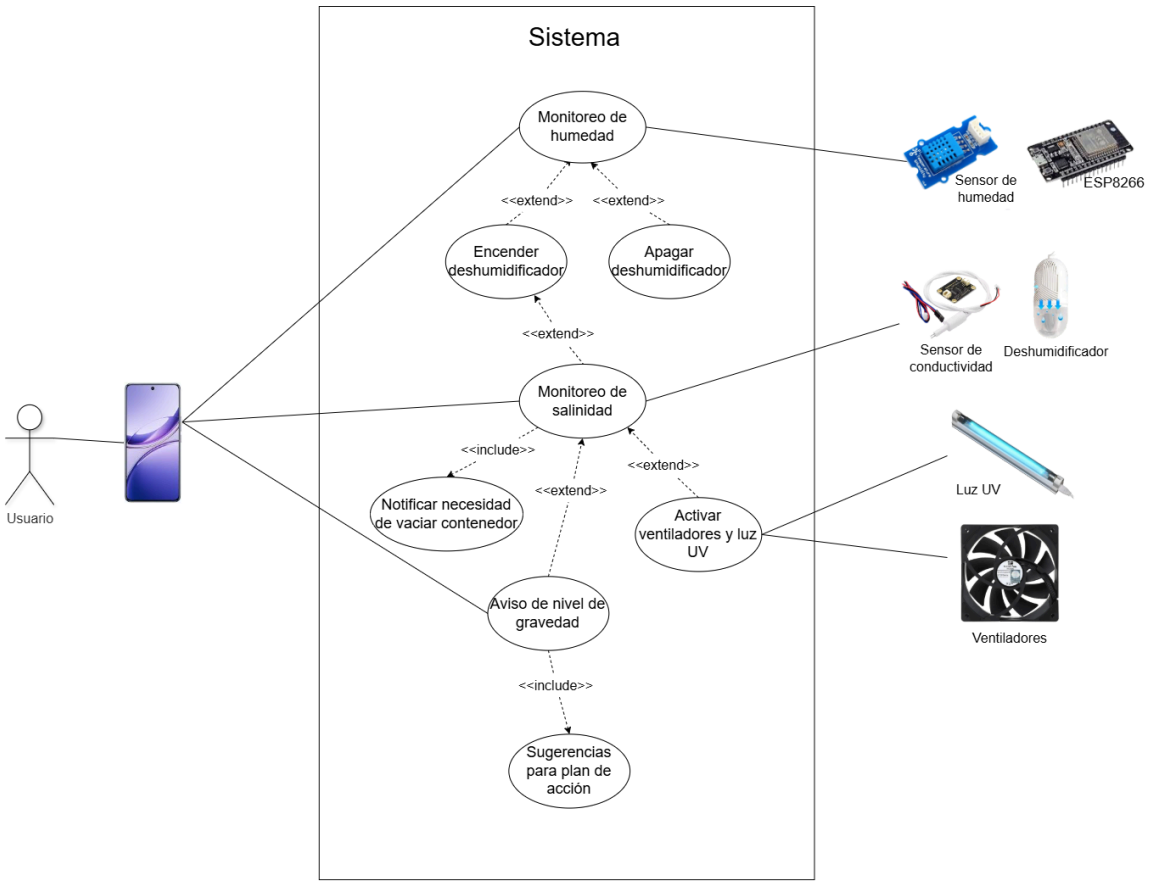


Figura 2. Casos de uso general (fuente propia).

Tabla 8. Caso de Uso: Monitoreo de humedad

ID: CU-001	
Nombre del CU: Monitoreo de humedad	
Actor(es): Sensor de humedad, usuario	
Requisitos Relacionados: RF02	
Descripción: El sistema recibe periódicamente las mediciones de humedad desde el sensor, las procesa y las muestra al usuario en la aplicación móvil. Si la humedad supera el umbral definido, se prende el deshumidificador. De lo contrario lo apaga.	
Precondiciones: El sensor de humedad debe estar conectado al Raspberry Pi 4B	
Flujo Principal: Usuario 1. Ingresa a la sección de monitoreo. 4. Visualiza el nivel de humedad.	Flujo Principal: Sistema 2. Reúne datos del sensor de humedad. 3. Muestra los datos en la interfaz web. 4. Evalúa si la humedad está dentro del rango normal. 5. [Extend: CU-002] Encender deshumidificador. (Opcional) 6. Guarda la lectura en la base de datos.
Flujo Principal: Usuario	Flujo Principal: Sistema

	5.1. [Extend: CU-003] Apagar deshumidificador. (Opcional) 6.1. Guarda la lectura en la base de datos.
Flujo Alternativo: Usuario 4.1 Recibe una alerta indicando que no se pudieron recibir los datos.	Flujo Alternativo: Sistema 2.1 No se logra comunicación con el sensor. 3.1 Envía una alerta debido a la desconexión entre el Raspberry Pi 4B y la aplicación.
Postcondiciones: ● El usuario ha visualizado correctamente el nivel de humedad actual. ● Si hubo una falla, el sistema notifica al usuario mediante una alerta. ● El sistema queda en espera para la próxima lectura automática.	
Reglas de Negocio: ● El sistema debe actualizar los datos de humedad cada 5 segundos. ● Si no se reciben datos en un intervalo de 10 segundos, se debe generar una alerta. ● Las alertas deben ser visibles en la interfaz del usuario en menos de 2 segundos tras detectarse el error.	
CU Relacionados: Extend CU-002 (Prender deshumidificador)	

Tabla 9. Caso de Uso: Encender deshumidificador

ID: CU-002	
Nombre del CU: Encender deshumidificador	
Actor(es): Deshumificador	
Requisitos Relacionados: RF03	
Descripción: El sistema activa automáticamente el deshumidificador cuando detecta que el nivel de humedad supera el umbral permitido. Si se llega a la recolección de agua necesaria se procede a realizar el monitoreo de salinidad.	
Precondiciones: El deshumidificador debe estar conectado al Raspberry Pi 4B. El sistema recibió la lectura de humedad. El recipiente del humidificador debe estar vacío.	
Flujo Principal: Deshumidificador 3. Captura el agua condensada del ambiente cerrado	Flujo Principal: Sistema 1. Detecta niveles de humedad que sobrepasan los umbrales configurados. 2. Envía señal para encender el deshumidificador 4. Registra el estado del deshumidificador. 5. Monitorea continuamente la cantidad de agua recolectada en el contenedor.

	6. [Extend CU-004] Monitoreo de salinidad. (Opcional)
Flujo Alternativo: Usuario 4.1. Recibe una alerta indicando que no se pudo encender el deshumidificador.	Flujo Alternativo: Sistema 2.1. No se logra conectar con el deshumidificador. 3.1. El sistema envía una alerta debido a la desconexión entre el Raspberry Pi 4B y el activador del deshumidificador 5.1. Intenta reconectar
Postcondiciones: ● El agua recolectada ha superado el umbral mínimo definido (por ejemplo, 100 ml) ● El deshumidificador queda encendido.	
Reglas de Negocio: ● El sistema debe activar el deshumidificador si la humedad supera el umbral configurado. ● El sistema debe verificar cada minuto si la cantidad de agua recolectada ha alcanzado el umbral mínimo.	
CU Relacionados : Extend CU-004 (Monitoreo de salinidad)	

Tabla 10. Caso de Uso: Apagar deshumidificador

ID: CU-003	
Nombre del CU: Apagar deshumidificador	
Actor(es): Deshumidificador	
Requisitos Relacionados: RF06	
Descripción: El sistema activa automáticamente el deshumidificador cuando detecta que el nivel de humedad ha bajado entrando en el umbral permitido.	
Precondiciones: El deshumidificador debe estar conectado al Raspberry Pi 4B. El sistema recibió la lectura de humedad.	
Flujo Principal: Deshumidificador 3. Se apaga.	Flujo Principal: Sistema 1. Detecta niveles de humedad que ya no sobrepasan los umbrales configurados. 2. Envía señal para apagar el deshumidificador. 4. Registra el estado del deshumidificador.
Flujo Alternativo: Usuario 4.1. Recibe una alerta indicando que no se pudo apagar el deshumidificador.	Flujo Alternativo: Sistema 2.1. No se logra conectar con el deshumidificador. 3.1. El sistema envía una alerta debido a la desconexión entre el Raspberry Pi 4B y el activador del deshumidificador 5.1. Intenta reconectar
Postcondiciones: El deshumidificador queda apagado.	

Reglas de Negocio: ● El sistema debe activar el deshumidificador si la humedad no supera el umbral configurado.
CU Relacionados : Extend CU-004 (Activar ventiladores y luz UV)

Tabla 11. Caso de Uso: Monitoreo de salinidad

ID: CU-004	
Nombre del CU: Monitoreo de salinidad	
Actor(es): Sensor de salinidad, deshumidificador, usuario	
Requisitos Relacionados: RF02	
Descripción: El sistema mide la salinidad del contenedor de agua y muestra los valores al usuario. Si se detecta que hay agua en el contenedor del deshumidificador, se notifica la necesidad de vaciar el contenedor. Si se detecta que la salinidad es alta se activan las medidas de control. Si siguen sin bajar los niveles, luego de las medidas de control, se avisa el nivel de gravedad.	
Precondiciones: El sensor de salinidad debe estar conectado al Raspberry Pi 4B. El deshumidificador debe tener el agua necesaria para medir la salinidad.	
Flujo Principal: Usuario 2. Ingresa a la sección de monitoreo. 5. Visualiza el nivel de salinidad.	Flujo Principal: Sistema 1. [Extend: CU-005] Notificar necesidad de vaciar contenedor. 3. Reúne datos del sensor de salinidad. 4. Envía los datos a la aplicación. 5. Evalúa si la salinidad está dentro del rango normal. 6. [Extend: CU-006] Activar ventiladores y luz UV (Opcional)
Flujo Alternativo: Usuario	Flujo Alternativo: Sistema 6.1. [Extend: CU-007] Avisar de nivel de gravedad (Opcional)
Flujo Alternativo: Usuario 4.1 Recibe una alerta de que no se pudieron recibir los datos.	Flujo Alternativo: Sistema 3.1 El sistema envía una alerta debido a la desconexión entre el Raspberry Pi 4B y la aplicación.
Postcondiciones: ● El usuario ha visualizado correctamente el nivel de salinidad actual. ● Si hubo una falla, el sistema notifica al usuario mediante una alerta.	
Reglas de Negocio: ● Si no se reciben datos en un intervalo de 10 segundos, se debe generar una alerta. ● Las alertas deben ser visibles en la interfaz del usuario en menos de 2 segundos tras detectarse el error.	
CU Relacionados : Include: CU-005 (Notificar necesidad de vaciar contenedor), Extend: CU-006 (Activar ventiladores y luz UV (Opcional)), Extend: CU-007 (Avisar de nivel de gravedad (Opcional))	

Tabla 12. Caso de Uso: Notificar necesidad de vaciar contenedor.

ID: CU-005	
Nombre del CU: Notificar necesidad de vaciar contenedor.	
Actor(es): Usuario, humidificador.	
Requisitos Relacionados: RF10	
Descripción: El sistema detecta que el humidificador contiene agua que debe ser retirada ya que se necesitan tomar nuevas medidas. Se le notifica al usuario vaciar el contenedor del deshumidificador.	
Precondiciones: El humidificador debe estar conectado al Raspberry Pi 4B	
Flujo Principal: Usuario 5. Visualiza la notificación. 6. Selecciona el botón de finalizar una vez terminado el proceso.	Flujo Principal: Sistema 1. Verifica si no es la primera medición 2. Verifica el nivel de agua del contenedor del deshumidificador. 3. Determina si el nivel es suficiente como para requerir el vaciado del contenedor. 4. Muestra la notificación en la interfaz web. 7. Verifica que el contenedor del deshumidificador está vacío.
Flujo Alternativo: Usuario 4.1 Recibe una alerta de que no se vacío el contenedor correctamente.	Flujo Alternativo: Sistema 7.1. Notifica que el contenedor sigue teniendo agua.
Postcondiciones: El sistema habilita la siguiente medición de salinidad.	
Reglas de Negocio:	
CU Relacionados :	

Tabla 13. Caso de Uso: Activar ventiladores y luz UV

ID: CU-006	
Nombre del CU: Activar de ventiladores y luz UV	
Actor(es): Ventiladores, luz UV, usuario	
Requisitos Relacionados: RF09	
Descripción: Cuando se realiza la primera medición de salinidad y esta supera el umbral definido. El sistema da la posibilidad de encender las medidas de control: los ventiladores, luz UV.	
Precondiciones: Los ventiladores y la luz UV deben estar conectados al Raspberry Pi 4B	
Flujo Principal: Usuario	Flujo Principal: Sistema

4. Ve la notificación y acepta que se activen las medidas. 8. Visualiza la notificación de confirmación.	1. Detecta que la salinidad está por encima del umbral configurado. 2. Muestra una notificación a la web indicando que se quieren activar las medidas de control. 5. Enciende los ventiladores, la luz UV. 6. Registra la activación. 7. Envía la notificación de que las medidas fueron activadas correctamente.
Flujo Alternativo: Usuario 4.1. Ve la notificación y aplaza la activación de las medidas. 8.1. Visualiza la notificación de confirmación para el aplazo.	Flujo Alternativo: Sistema 5.1. Registra que la acción fue aplazada. 5.2. Programa la activación de las medidas luego de 5 minutos. 5.3. Notifica al usuario que la acción ha sido aplazada.
Postcondiciones:	
Reglas de Negocio:	
CU Relacionados :	

Tabla 14. Caso de Uso: Aviso de nivel de gravedad

ID: CU-007	
Nombre del CU: Aviso de nivel de gravedad	
Actor(es): Usuario	
Requisitos Relacionados: RF12	
Descripción: Envía al usuario una notificación indicando el nivel de gravedad detectada.	
Precondiciones: El sistema recibió la lectura de salinidad y humedad.	
Flujo Principal: Usuario 5. Visualiza la notificación. 6. Ingresa a la sección de niveles de gravedad.	Flujo Principal: Sistema 1. Evalúa los datos provenientes de los sensores. 2. Determinar si los valores están fuera del rango normal. 3.Determinar de qué nivel de gravedad se trata, acumulado con días anteriores. 4. Si el nivel corresponde a un estado grave, el sistema genera un aviso.

	7. [Extend CU-008] Sugerencias para plan de acción
Flujo Alternativo: Usuario	Flujo Alternativo: Sistema 4.1. Si no es suficientemente grave, almacena el valor y no manda aviso.
Postcondiciones: El usuario ha visualizado correctamente el aviso entregado por notificación.	
Reglas de Negocio:	
CU Relacionados : Extend CU-008 (Sugerencias para plan de acción)	

Tabla 15. Caso de Uso: Sugerencias para plan de acción

ID: CU-008	
Nombre del CU: Sugerencias para plan de acción	
Actor(es): Usuario	
Requisitos Relacionados: RF13	
Descripción: Proporciona recomendaciones concretas cuando las medidas de control (ventiladores, luz UV, deshumidificador) no logran reducir la humedad o la salinidad a niveles seguros.	
Precondiciones: Se debe de haber medido el nivel de gravedad.	
Flujo Principal: Usuario 1. Ingresa a la sección de sugerencias seleccionando el nivel de gravedad. 5. Visualiza las sugerencias.	Flujo Principal: Sistema 2. Identifica el nivel de gravedad y la mediciones de los sensores. 3. Genera las recomendaciones 4. Envía las recomendaciones a la web.
Flujo Alternativo: Usuario 5.1 Recibe una alerta de que no se pudieron recibir los datos.	Flujo Alternativo: Sistema 4.1 El sistema envía una alerta debido a la desconexión entre el Raspberry Pi 4B y la aplicación. 5.2. Intenta reconectar.
Postcondiciones: Se almacena un registro de la recomendación emitida.	
Reglas de Negocio:	
CU Relacionados :	

4.1.1.3. Descripción de la Arquitectura (vista del modelo diseño)

Este diagrama de arquitectura representa el flujo de datos y control en un sistema automatizado de gestión ambiental. El núcleo del sistema es una Raspberry Pi (1), que actúa como la unidad central de procesamiento y control de decisiones. Esta unidad recibe información del entorno a través de un Sensor de humedad conectado a un módulo ESP8266 y un Sensor de conductividad (2). Basándose en las lecturas de estos sensores,

la Raspberry Pi activa diversos actuadores: una lámpara de Luz UV, Ventiladores y un Deshumidificador (3) para corregir las condiciones ambientales. Finalmente, el estado del sistema y las alertas se envían al Usuario (4), permitiendo la supervisión remota del proceso.

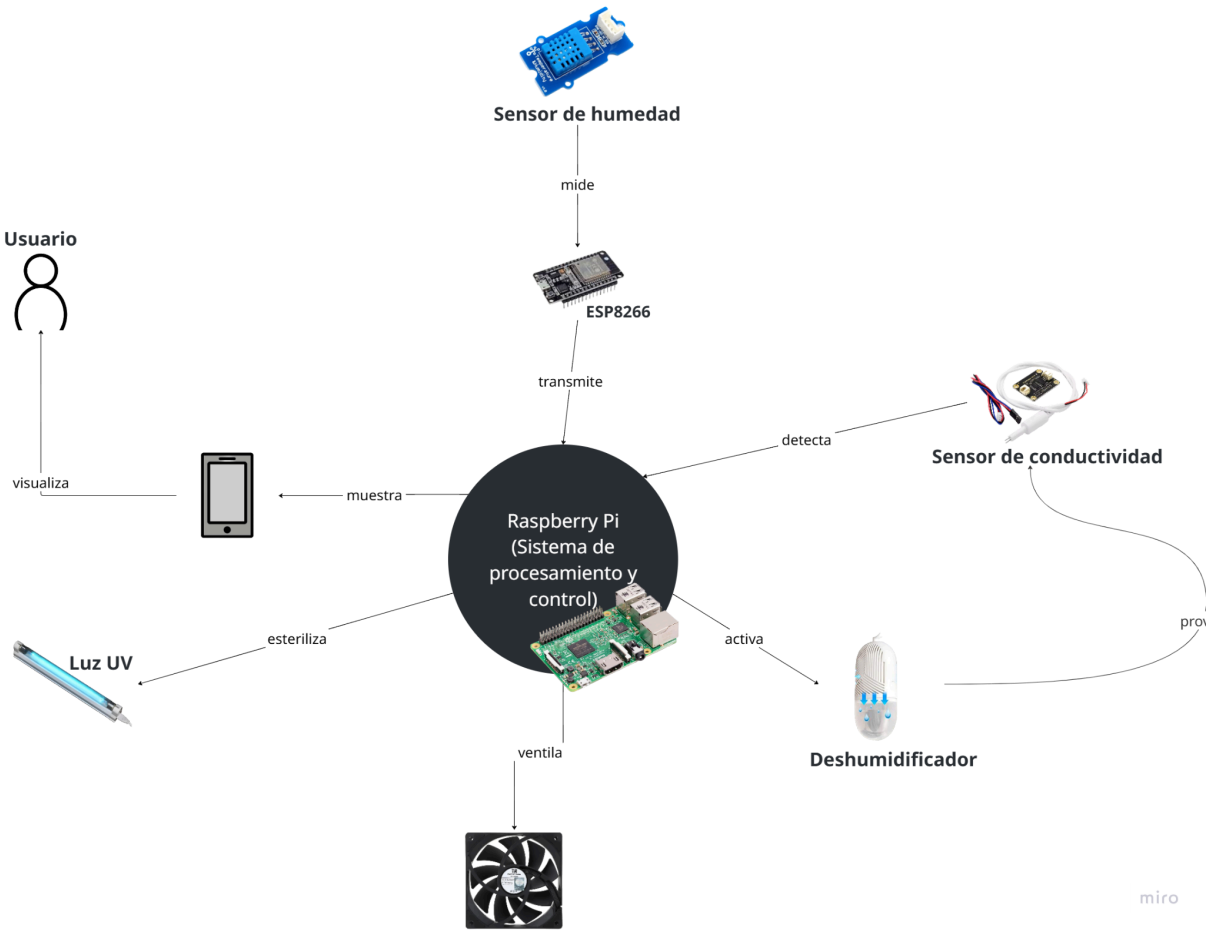


Figura 3. Diagrama de contexto.

Ventiladores: Son los encargados de la circulación de aire. Cuando el sistema detecta humedad alta, estos se activan para expulsar el aire viciado y húmedo y mover aire nuevo, evitando que la humedad se estanque en las paredes.

Luz UV: Es el sistema de desinfección. Funciona emitiendo radiación ultravioleta dirigida a la zona afectada para matar las esporas de moho y bacterias que proliferan con la humedad, impidiendo que crezcan colonias de hongos.

Deshumidificador: Es el recolector de agua. Su trabajo es condensar la humedad del aire (convertir el vapor en agua líquida) y almacenarla en un contenedor. Esta agua recolectada es vital porque es la que luego se usará para medir la sal.

Sensor de conductividad: Es el que detecta la sal. Se sumerge en el agua recolectada por el deshumidificador. Si el agua tiene mucha sal (salitre), conducirá mejor la electricidad. Este sensor mide esa capacidad para confirmar si hay presencia de sales en el ambiente.

ESP8266: Es el "ayudante inalámbrico". Es un microcontrolador pequeño que se conecta al sensor de humedad y envía esos datos vía Wi-Fi hacia la Raspberry Pi. Permite poner el sensor lejos del cerebro central sin usar cables largos.

Sensor de humedad: Es el termómetro del ambiente. Monitorea constantemente el porcentaje de Humedad Relativa (HR) en el aire de la habitación. Envía estos datos numéricos al controlador.

Raspberry Pi 4B: Es el cerebro central (CPU). Recibe toda la información del ESP8266 y de los sensores. Además, aloja la base de datos y la página web.

4.1.1.4. Documento de Diseño de Interfaz Usuario

El diseño de la interfaz de usuario (UI) se centra en ser claro e intuitivo para el usuario final. Está optimizado para visualización móvil y se organiza en cinco vistas principales que se acceden desde un menú de navegación inferior constante.
A continuación, se detalla la función de cada pantalla y su relación con los requisitos del sistema:

1. Vista Principal: Control de Humedad

Figura 4. Vista principal

Esta pantalla sirve como el punto de interacción principal con los actuadores del sistema. Presenta un panel con interruptores para el Ventilador, la Luz UV y el Deshumidificador. Aunque el sistema está diseñado para activar estos dispositivos automáticamente (RF09), esta vista permite al usuario tener un control manual y la opción de desactivar o posponer las medidas de control (RF08) con solo presionar un botón, asegurando que las acciones correctivas se puedan gestionar según las necesidades de el ambiente.

2. Vista: Niveles de Humedad

Figura 5. Niveles de humedad

Esta es la vista de monitoreo en tiempo real, donde el usuario recibe la información esencial del ambiente. Aquí se cumplen los requisitos de mostrar la Humedad Relativa y la Salinidad .Cada medición se muestra en un recuadro claro con un indicador de "Actualizado", confirmando la recepción exitosa de los datos provenientes de la Raspberry Pi. La frecuencia de actualización de los datos de humedad está definida para ser cada 5 segundos, garantizando una vigilancia constante.

3. Vista: Gravedad

Figura 6. Gravedad

La pantalla de Gravedad tiene la función de proporcionar un diagnóstico visual instantáneo y el historial de la situación ambiental. La sección de "Último nivel de gravedad" utiliza un llamativo recuadro Rojo para resaltar inmediatamente una condición "Alto", cumpliendo con el requisito de avisar el nivel de gravedad. Además, incluye una Gráfica Semanal que utiliza barras azules para ilustrar la evolución de la humedad y la salinidad, permitiendo al usuario consultar el historial de lecturas y visualizar el progreso del sistema.

4. Vista: Recomendaciones

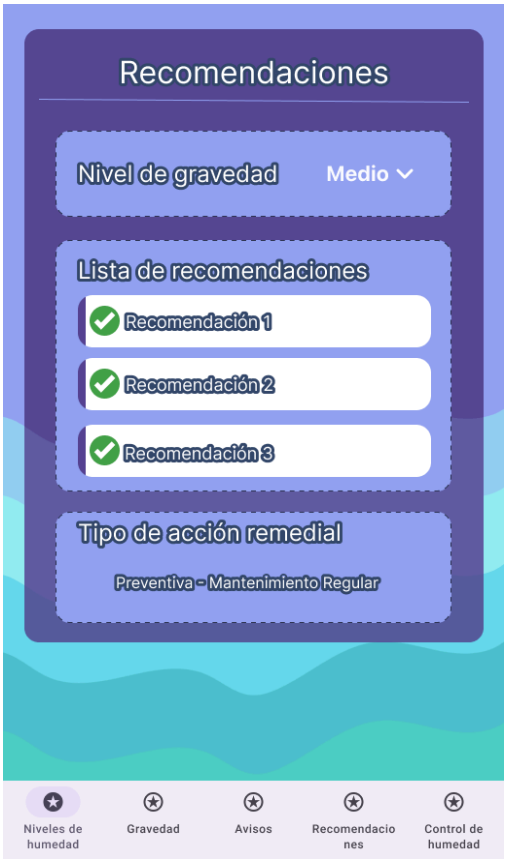


Figura 7. Recomendaciones

Esta vista se enfoca en la acción y solución, activándose después de que el sistema evalúa que las medidas de control no han sido suficientes. Permite al usuario filtrar por Nivel de gravedad y muestra una Lista de recomendaciones con marcas de verificación Verdes, lo que indica que son consejos procesados y accionables. Finalmente, se especifica el tipo de acción remedial, asegurando que el usuario reciba sugerencias concretas para reducir el riesgo.

5. Vista: Avisos

Figura 8. Avisos

La pantalla de Avisos actúa como el registro de todas las notificaciones y alertas del sistema. El diseño utiliza un claro icono de Campana en cada aviso, lo que dirige la atención del usuario de manera inmediata. Esta sección es crucial para los requisitos de notificación, ya que el sistema envía alertas aquí cuando se detectan altos niveles de sal o cuando el deshumidificador necesita ser vaciado. La información debe aparecer con una latencia de menos de 2 segundos tras la detección.

4.1.1.5. Modelo de Diseño (Diagrama de clases)

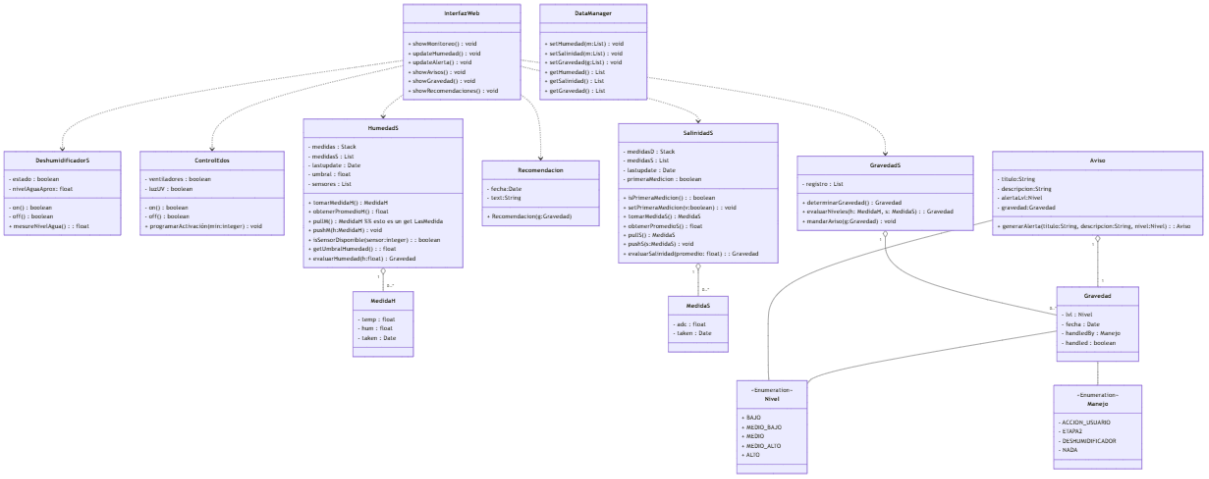


Figura 8. Diagrama de clases

4.1.1.6. Modelo de Interacción (Diagrama de secuencia, o flujo de las acciones)

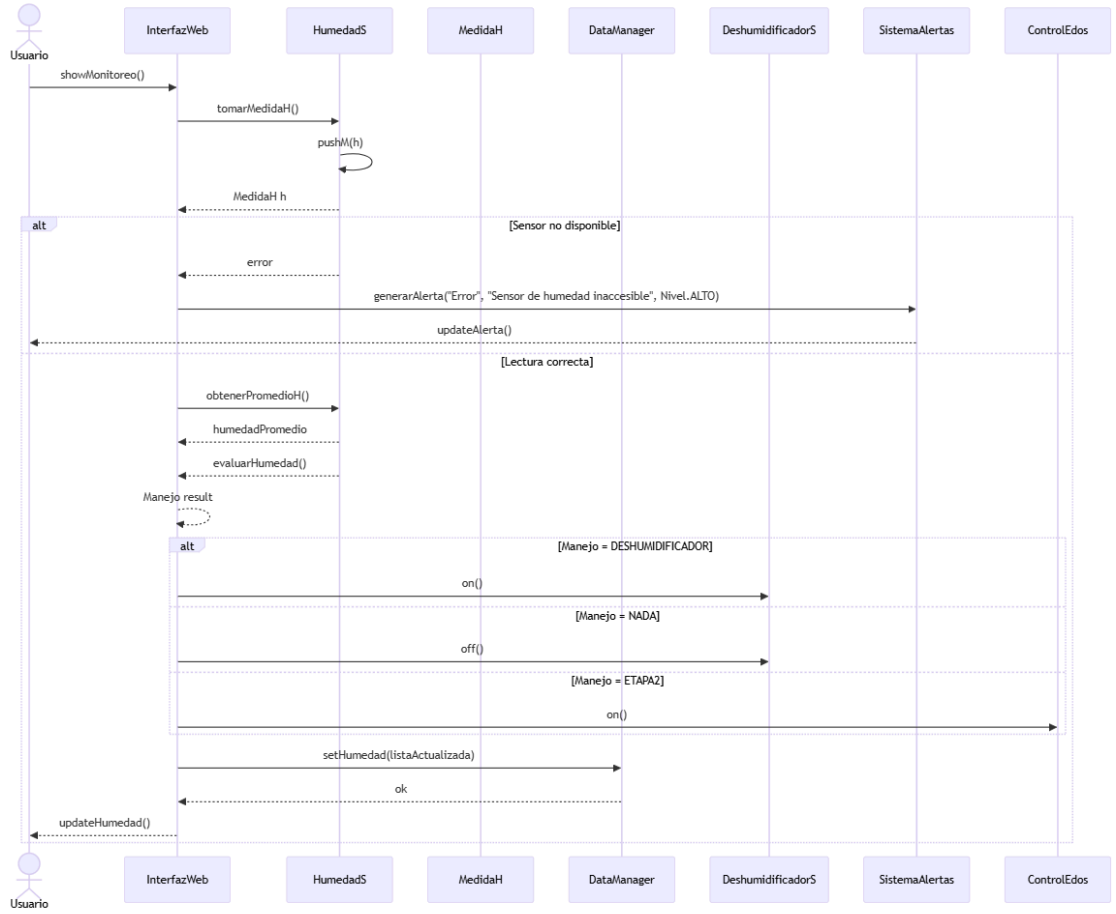


Figura 9. CU-001 Monitoreo de humedad, Diagrama de secuencia.

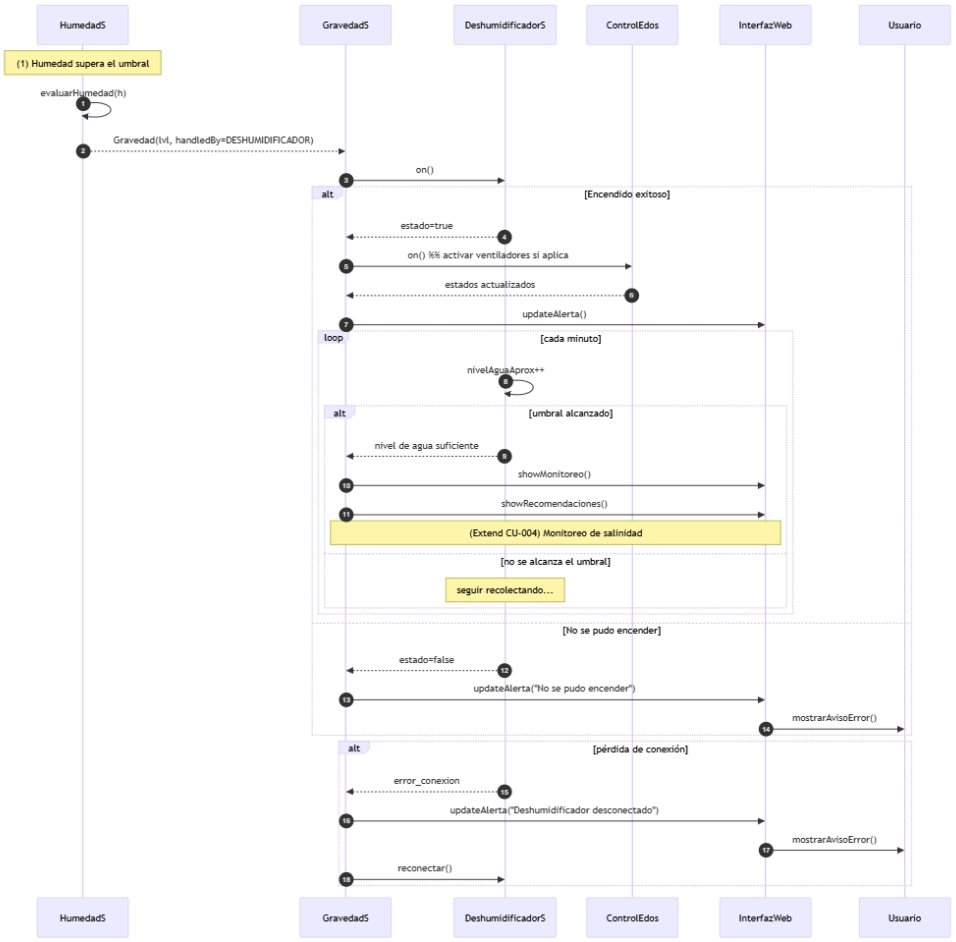


Figura 10. CU-002 Encender deshumidificador, Diagrama de secuencia

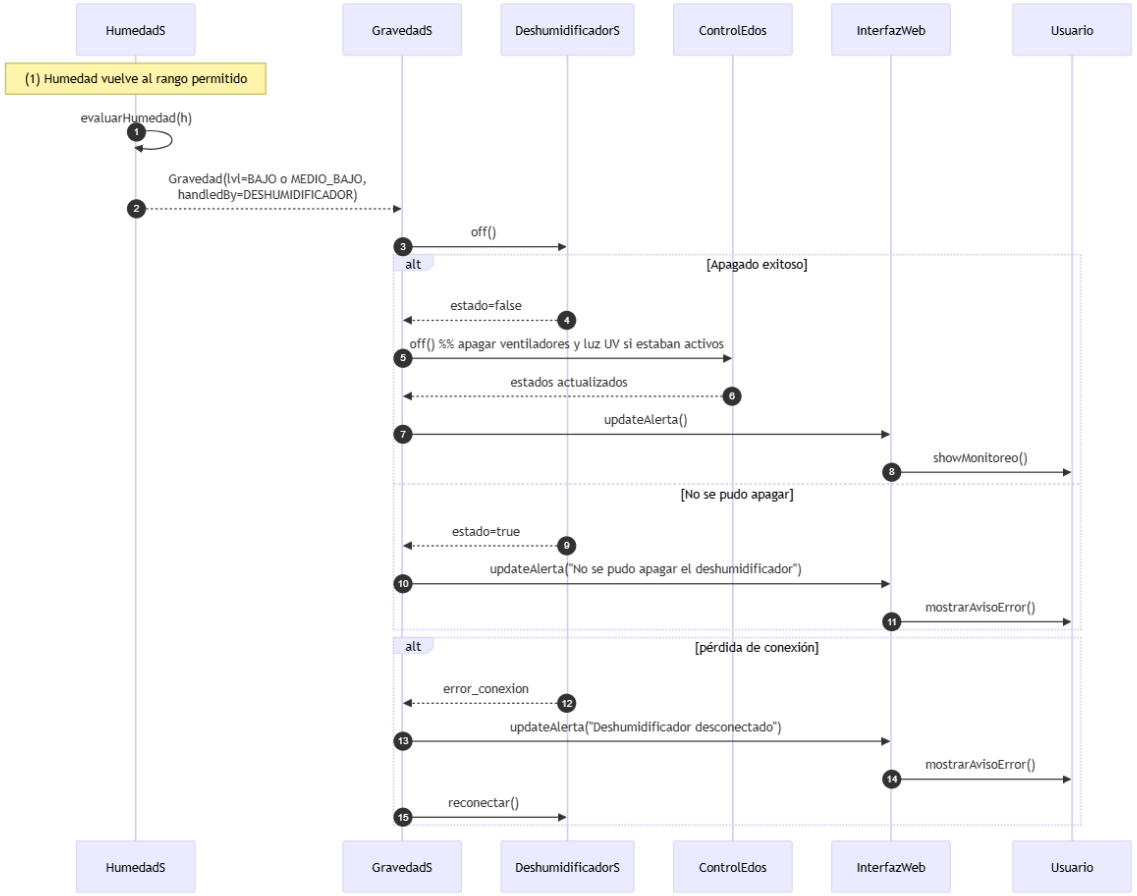


Figura 11. CU-003 Apagar deshumidificador, Diagrama de secuencia

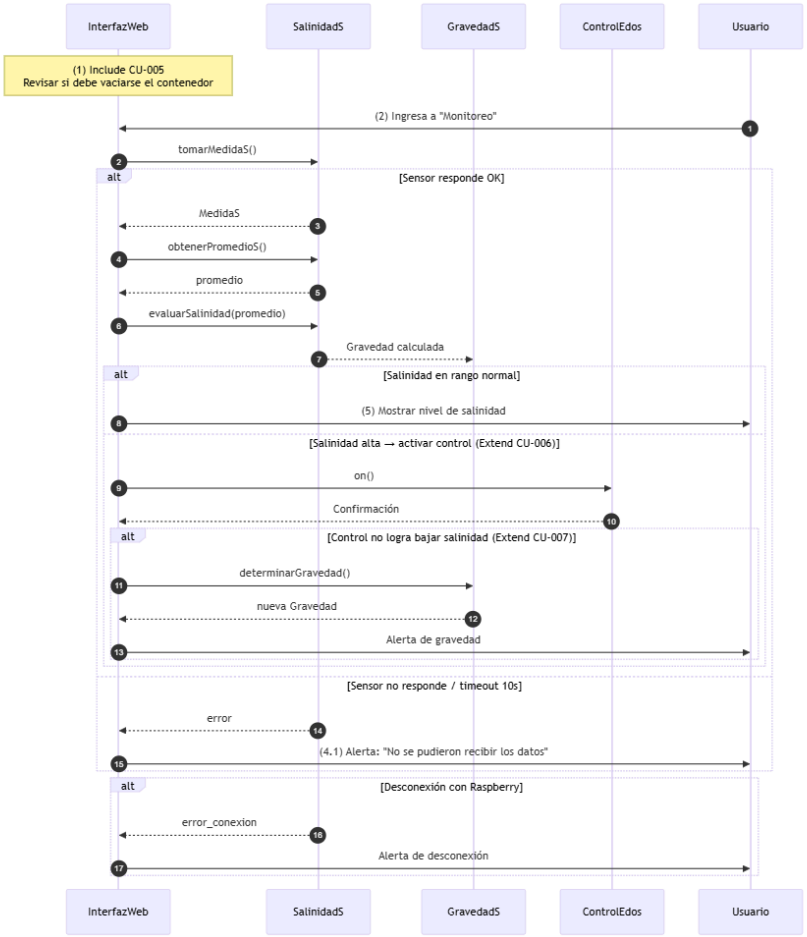


Figura 12. CU-004 Monitoreo de salinidad, Diagrama de secuencia

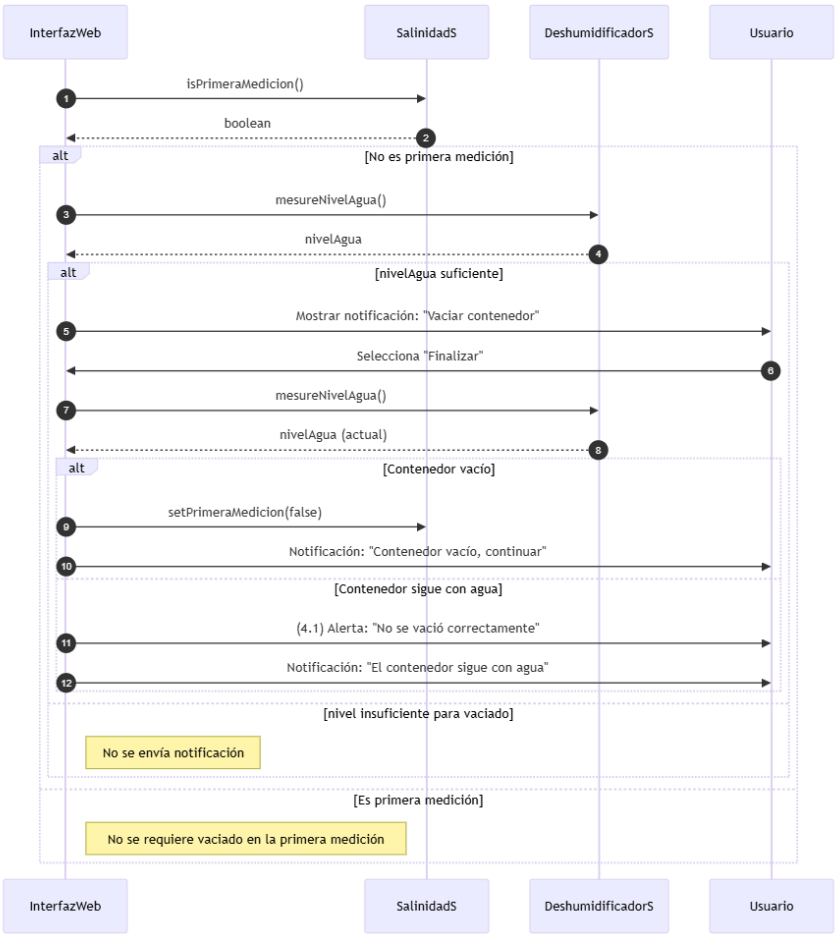


Figura 13. CU-005 Notificar necesidad de vaciar contenedor, Diagrama de secuencia

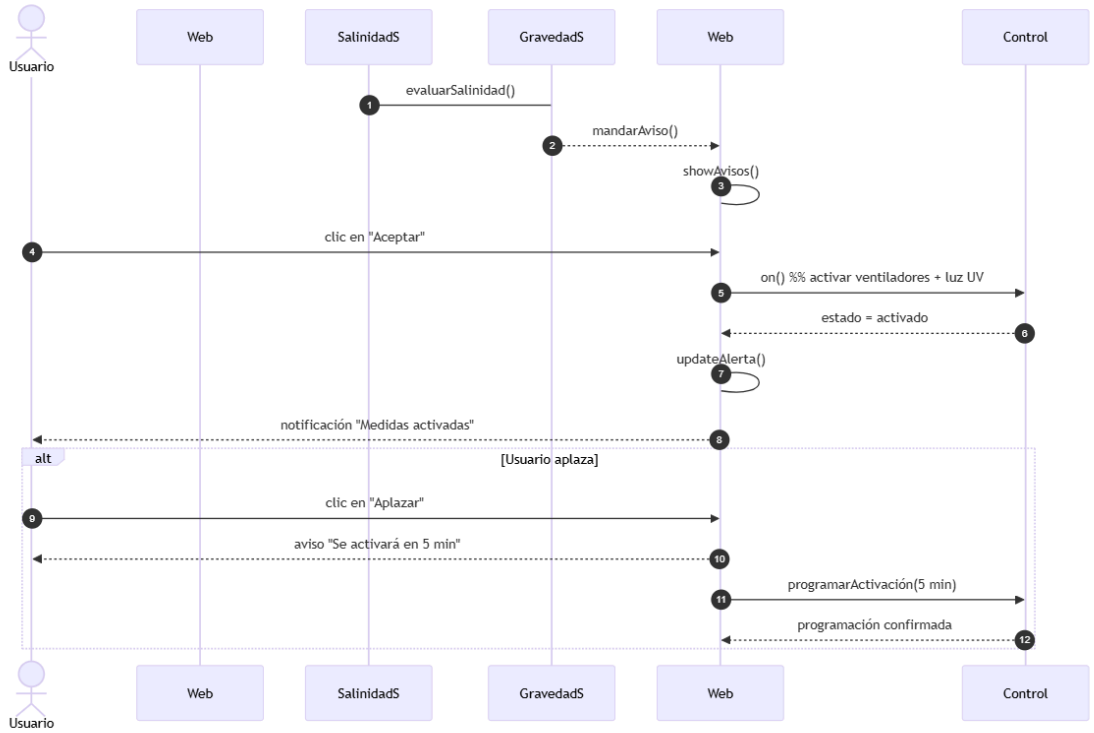


Figura 14. CU-006 Activar de ventiladores y luz UV, Diagrama de secuencia

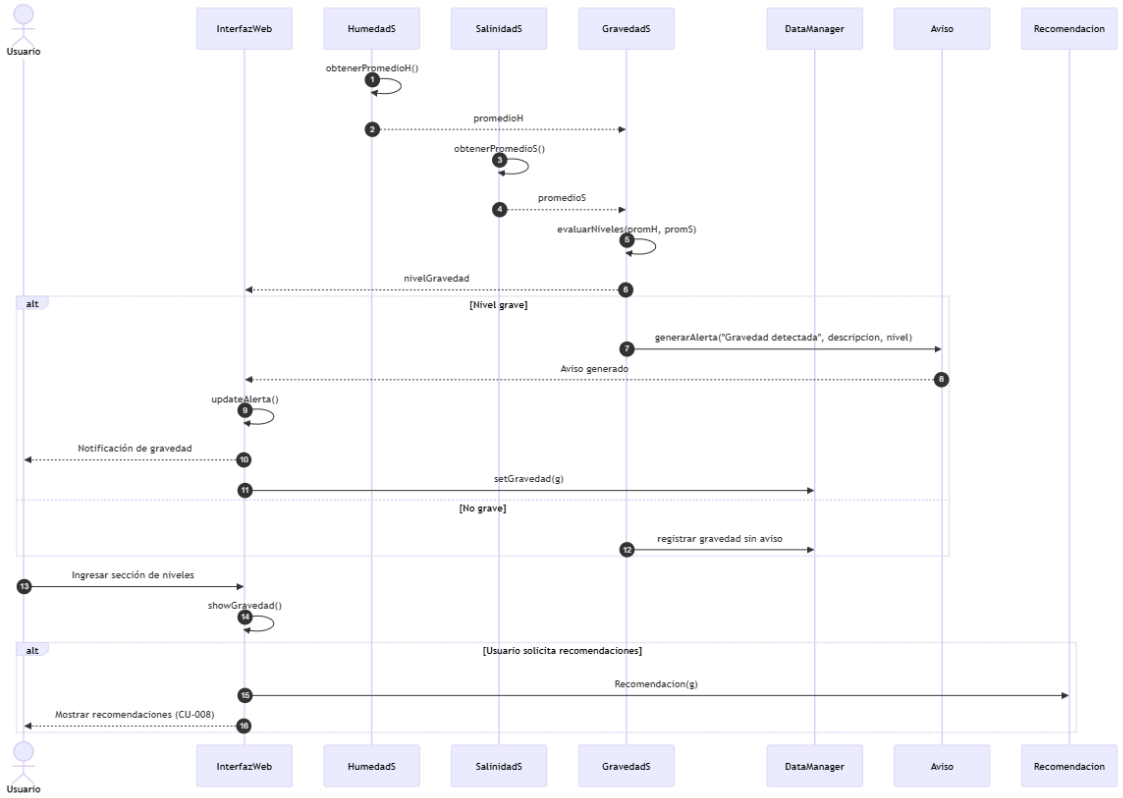


Figura 15. CU-007 Aviso de nivel de gravedad, Diagrama de secuencia

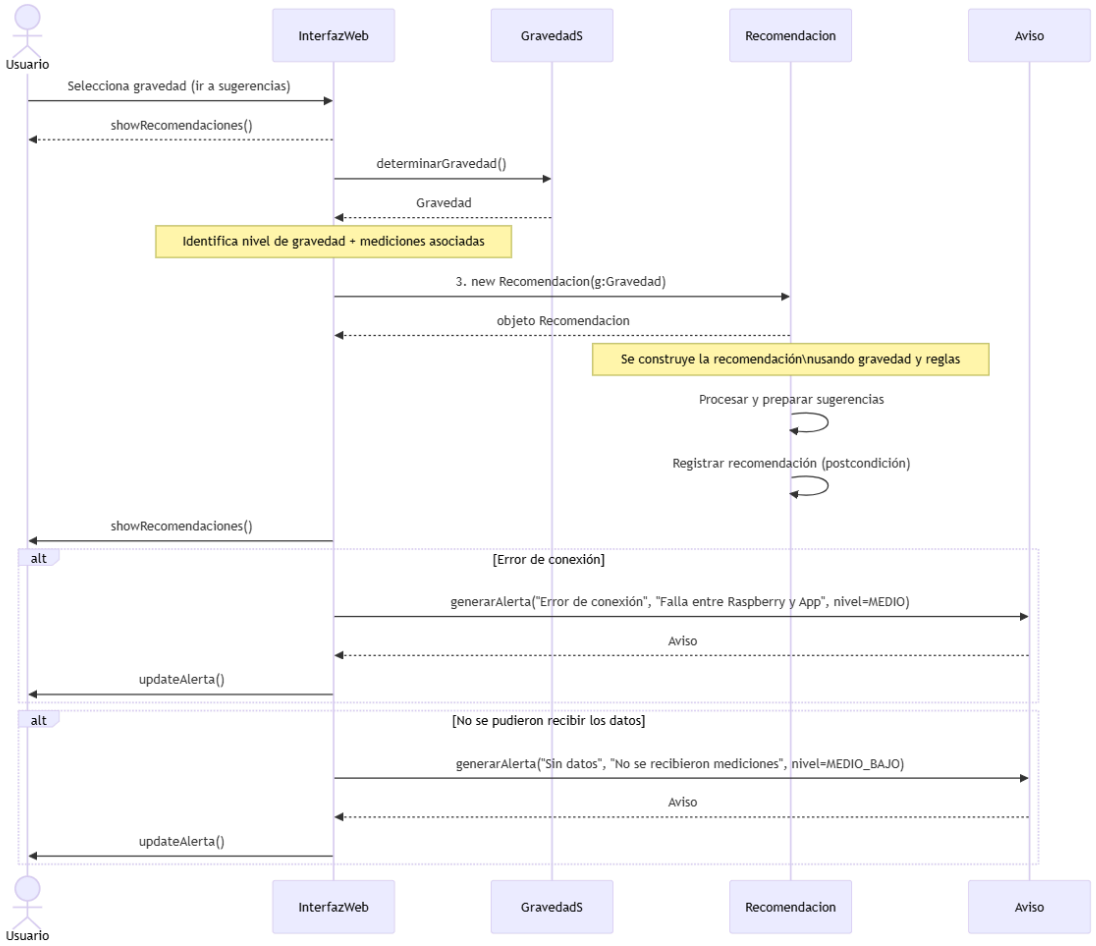


Figura 16. CU-008 Sugerencias para plan de acción, Diagrama de secuencia

4.1.1.7. Descripción de la Arquitectura vista del modelo diseño.

Figura

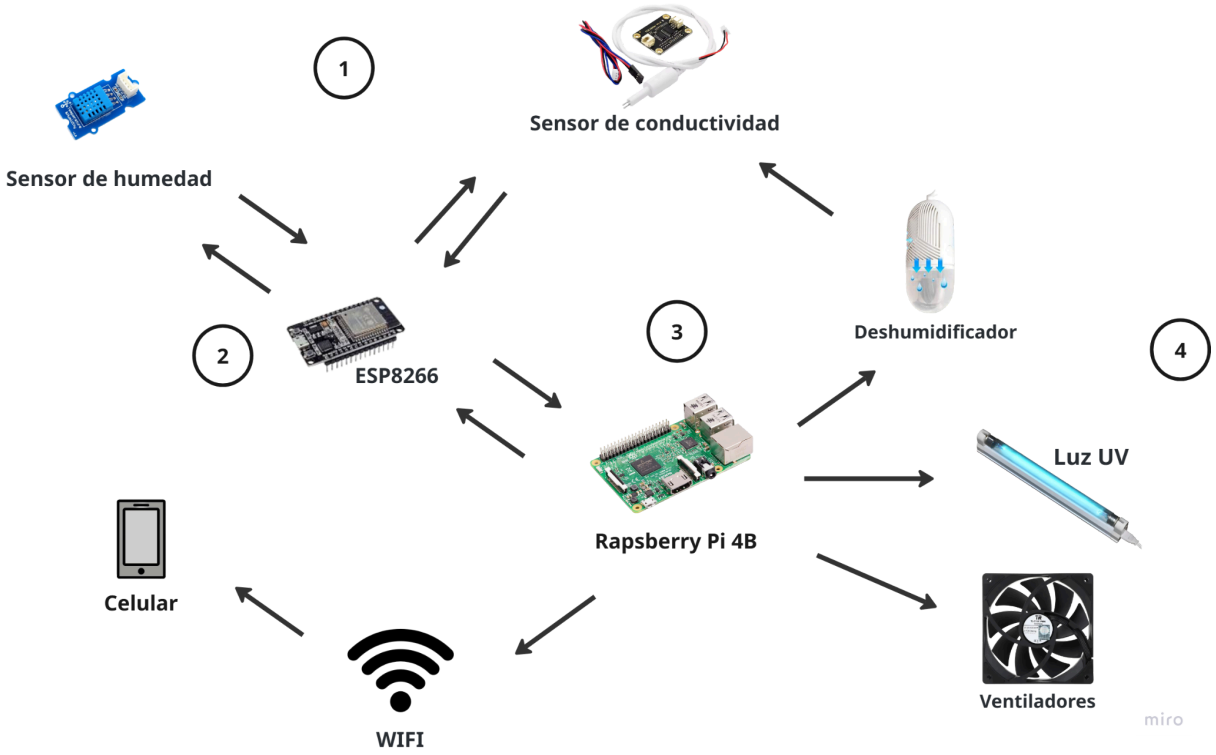


Figura 17. Arquitectura vista del modelo diseño.

La arquitectura del sistema se basa en un modelo IoT distribuido, compuesto por sensores, actuadores, un microcontrolador (ESP8266), un microprocesador (Raspberry Pi 4B), un backend de procesamiento y una interfaz web para el usuario final. Esta arquitectura permite la recolección de datos ambientales, su procesamiento y la activación de componentes físicos y la notificación de alertas de manera integrada y eficiente.

El sistema se organiza en tres niveles principales:

1. Nivel físico:
    a. Sensor de humedad.
    b. Sensor de conductividad (salinidad).
    c. Deshumidificador.
    d. Ventilación o ventilador.
    e. Luz UV.
    f. Microcontrolador (ESP8266).
    g. Microprocesador.
2. Nivel lógico:
    a. Módulo de sensores.
    b. Módulo de control.
    c. Módulo de procesamiento.
    d. Módulo de recomendaciones.
    e. Módulo de alertas.
3. Nivel de presentación:
    a. Mostrar mediciones de humedad y salinidad.
    b. Indicar estados del sistema.

c. Presentar alertas y notificaciones.
d. Permitir al usuario recibir sugerencias para acciones correctivas.
e. Diseño móvil responsive para fácil acceso.

Arquitectura de integración de los componentes:

1. Los sensores capturan datos
2. El ESP8266 procesa la humedad y envía los datos al microprocesador
3. La Raspberry Pi recibe la información y procesa la salinidad desde la conductividad en el sensor
4. El backend procesa la información, compara la tabla de gravedad y determina si se necesita la activación de actuadores, mandando la señal al microprocesador.
5. La interfaz web presenta visualización, alertas y sugerencias
6. El usuario recibe notificaciones y puede tomar acciones basadas en lo mostrado.

4.1.2. Implementación

4.1.2.1 Plan de Integración

Para asegurar que el sistema funcione correctamente, se utilizará una estrategia de integración progresiva por módulos. Esta estrategia permite unir y verificar cada componente de manera controlada, reduciendo el riesgo de fallas y facilitando la detección temprana de errores.

La integración se realizará en las siguientes etapas:

1. **Integración del módulo de sensores**: Se conectarán y validará los sensores de humedad y salinidad, asegurando que entreguen datos confiables y en el formato esperado.
2. **Integración del módulo de control**: Se conectarán y validará que los dispositivos que requieren encendido y apagado funcionen correctamente.
3. **Integración del módulo de procesamiento**: Los datos obtenidos por los sensores se integrarán con el módulo encargado de procesar las mediciones y compararlas con los umbrales definidos en la tabla de gravedad.
4. **Integración del módulo de recomendaciones**: Una vez determinado el módulo de gravedad, se integrará el módulo de recomendaciones, que según la gravedad, generará recomendaciones adecuadas para solucionar el problema, y disminuir la gravedad.
5. **Integración del módulo de alertas**: Se integrará el sistema de notificaciones que enviará avisos al usuario cuando se detecta una condición grave un cambio en el sistema físico.
6. **Integración de la interfaz de usuario**: Tras asegurar el funcionamiento interno del sistema, se conectará la interfaz web o aplicación con los módulos anteriores, permitiendo al usuario visualizar mediciones, recibir alertas y acceder a sugerencias de acción.
7. **Prueba de integración completa**: Finalmente, se realizará una prueba end-to-end, para validar que cada módulo interactúe correctamente con los demás, confirmando que el flujo completo, funcione sin errores.

Esta estrategia garantiza una integración ordenada y controlada del sistema, facilitando la identificación de errores y asegurando el funcionamiento estable.

4.1.2.2 Descripción de la Arquitectura (vista desde los módulos del caso de uso)

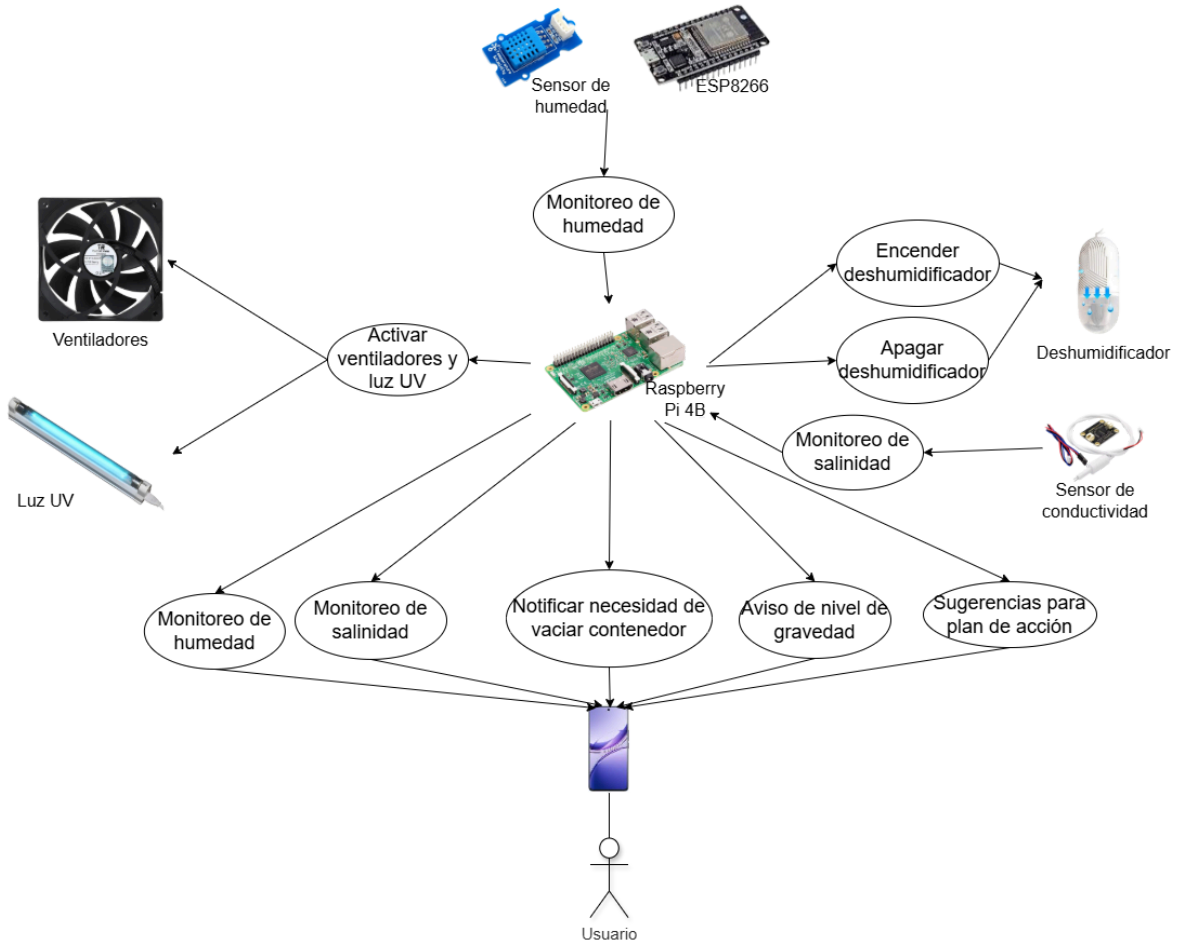


Figura 18. Arquitectura vista desde los módulos del caso de uso.

Componentes hardware:

1. Sensores:
    a. Sensor de humedad
    b. Sensor de conductividad
2. Actuadores:
    a. Deshumidificador
    b. Ventilador o sistema de ventilación
    c. Luz UV
3. Microcontrolador
    a. ESP8266: Enviar información a través de WiFi hacia el microprocesador.
4. Microprocesador
    a. Raspberry Pi 4 B

Componentes software:

1. Módulo de sensores: Implementado para casos de uso “Monitoreo de humedad” y “Monitoreo de salinidad”.
    a. Módulo de monitoreo de humedad: Implementado dentro del microcontrolador, se obtendrán los datos del sensor de humedad y se mandarán al microprocesador.
    b. Módulo de monitoreo de salinidad: Implementado dentro del microprocesador.
2. Módulo de control: Implementado para casos de uso “Encender/Apagar deshumidificador” y “Activar ventilación”
    a. Encender/Apagar deshumidificador
    b. Activar/Apagar Ventilación
3. Módulo de procesamiento:Comparará los valores leídos con la tabla de gravedad. Implementa la lógica de:
    a. Monitoreo de humedad
    b. Monitoreo de salinidad
    c. Determinación de nivel de gravedad
4. Módulo de recomendaciones: Implementado para el caso de uso “Sugerencias para plan de acción”
    a. Genera las sugerencias para plan de acción según el nivel de gravedad
5. Módulo de alertas: Implementado para el caso de uso “Aviso de nivel de gravedad” y “Notificar vaciar contenedor”
    a. Permite notificar al usuario
    b. Avisar de eventos físicos
6. Módulo de interfaz de usuario
    a. Una aplicación web, con interfaz diseñada para móvil que permite unificar todo este sistema para la visualización del usuario.

Arquitectura de integración

1. Los sensores capturan datos del entorno.
2. El microcontrolador y microprocesador procesa los datos.
3. Según la etapa del sistema, se activan actuadores o se generan alertas.
4. La información procesada se envía a la interfaz de usuario.
5. El usuario recibe notificaciones y sugerencias de acción.

4.1.2.3 Modelo de Implementación

El modelo de implementación describe la estructura técnica del sistema, especificando cómo se organizan los componentes de software y hardware necesarios para su funcionamiento. Este modelo detalla los módulos principales, su distribución en capas y la tecnología utilizada para su desarrollo y despliegue.

Arquitectura general:

1. Capa de dispositivos (IoT).
2. Capa de servidor/Backend.
3. Capa de Presentación (Interfaz web).

Estructura del código:

- Sensor:

    - Lectura de sensores y envío de datos
    - Encendido y apagado de periféricos.
- Backend:
    - Rutas: definición de endpoints REST.
    - Controladores: lógica del negocio.
    - Modelos: Interacción con los datos.
    - Servicios: Procesamiento de alertas.
- Frontend:
    - Vistas: pantallas de la interfaz.
    - Componentes: elementos reutilizables.
    - *Assets:*estilos e imágenes.

Tecnologías utilizadas:

- Microcontrolador: ESP8266.
- Microprocesador: Python.
- Backend: Node.js (Express), Python
- Frontend: Angular.
- Protocolo de comunicación: HTTP REST y SSH
- Herramientas: Git, VSCode.

4.1.2.4 Módulos Implementados

En la presente implementación del sistema se desarrollaron la totalidad de los módulos definidos en la etapa de diseño, cumpliendo con todos los casos de uso planteados a excepción del caso de uso 5 “*Notificar vaciar contenedor*”, donde no se logró determinar un cálculo efectivo para advertir sobre el tanque lleno, pero aquello no interviene en el funcionamiento, si no que se trata de una funcionalidad aparte.

A continuación, se describen los módulos implementados y sus respectivas funcionalidades:

- **Módulo de sensores:** Implementado para los casos de uso *“Monitoreo de humedad”* y *“Monitoreo de salinidad”*. Permite la adquisición de datos desde los sensores físicos conectados al sistema.

backend > cliente.py > ...

```python
def loop():
    while True:
        leer_sensor()
        leer_humedad()

        print(estado["humedad"])

        if estado["humedad"] != 0:
            guardar_humedad_conductividad(
                estado["humedad"].get("hum"),
                estado["humedad"].get("temp"),
                estado["humedad"].get("id"),
                estado["conductividad"]
            )

        print("Estado actual:", estado)
        print(DB_PATH)
        time.sleep(5)

if __name__ == "__main__":
    threading.Thread(target=loop, daemon=True).start()

    app.run(host="0.0.0.0", port=5000)
```

Figura 19. Cliente.py.

- **Módulo de monitoreo de humedad:** Implementado dentro del microcontrolador. Se encarga de obtener los datos del sensor de humedad y transmitirlos al microprocesador para su posterior procesamiento.

```
27  void loop() {
28
29    if (millis() - lastRead >= 2500) {
30      lastRead = millis();
31
32      float h = dht.readHumidity();
33      float t = dht.readTemperature();
34
35      if (isnan(h) || isnan(t)) {
36        Serial.println("DHT FAIL");
37        return;   // <-- NO enviamos nada
38      }
39
40      Serial.print("Temp: "); Serial.println(t);
41      Serial.print("Hum: "); Serial.println(h);
42
43      // Envío de datos reales
44      if (client.connect(host, httpPort)) {
45        String json = "{\"temp\": " + String(t) + ", \"hum\": " + String(h) + "}";
46
47        client.println("POST /data HTTP/1.1");
48        client.print("Host: "); client.println(host);
49        client.println("Content-Type: application/json");
50        client.print("Content-Length: "); client.println(json.length());
51        client.println();
52        client.print(json);
53      } else {
54        Serial.println("ERROR conectando");
55      }
56    }
57  }
```

Figura 20. Código monitoreo de la humedad en ESP8266.

- **Módulo de monitoreo de salinidad:** Implementado dentro del microprocesador. Permite la lectura y gestión de los valores de salinidad obtenidos desde los sensores correspondientes.

```python
# Conductividad
cond_port = 0  # A0
conductividad = 0

# Sensor tanque lleno (relé aislado)
tank_pin = 2   # D2
grovepi.pinMode(tank_pin, "INPUT")
tanque_lleno = False

# Relé deshumidificador
relay_pin = 4  # D4
grovepi.pinMode(relay_pin, "OUTPUT")
grovepi.digitalWrite(relay_pin, 0)  # apagado inicial


def read_conductividad():
    global conductividad
    while True:
        try:
            conductividad = grovepi.analogRead(cond_port)
        except:
            pass
        time.sleep(3)

def read_tanque():···

@app.route("/sensorC", methods=["GET"])
def get_conductividad():···

@app.route("/tanque", methods=["GET"])
def get_tanque():···

@app.route("/onDesH", methods=["POST"])
def on_desH():···
```

Figura 21. Código salinidad en Raspberry.

- **Módulo de control:** Implementado para los casos de uso *“Encender/Apagar deshumidificador”* y *“Activar/Apagar ventilación”*. Permite el control de los actuadores del sistema, específicamente el encendido y apagado del deshumidificador y del sistema de ventilación.

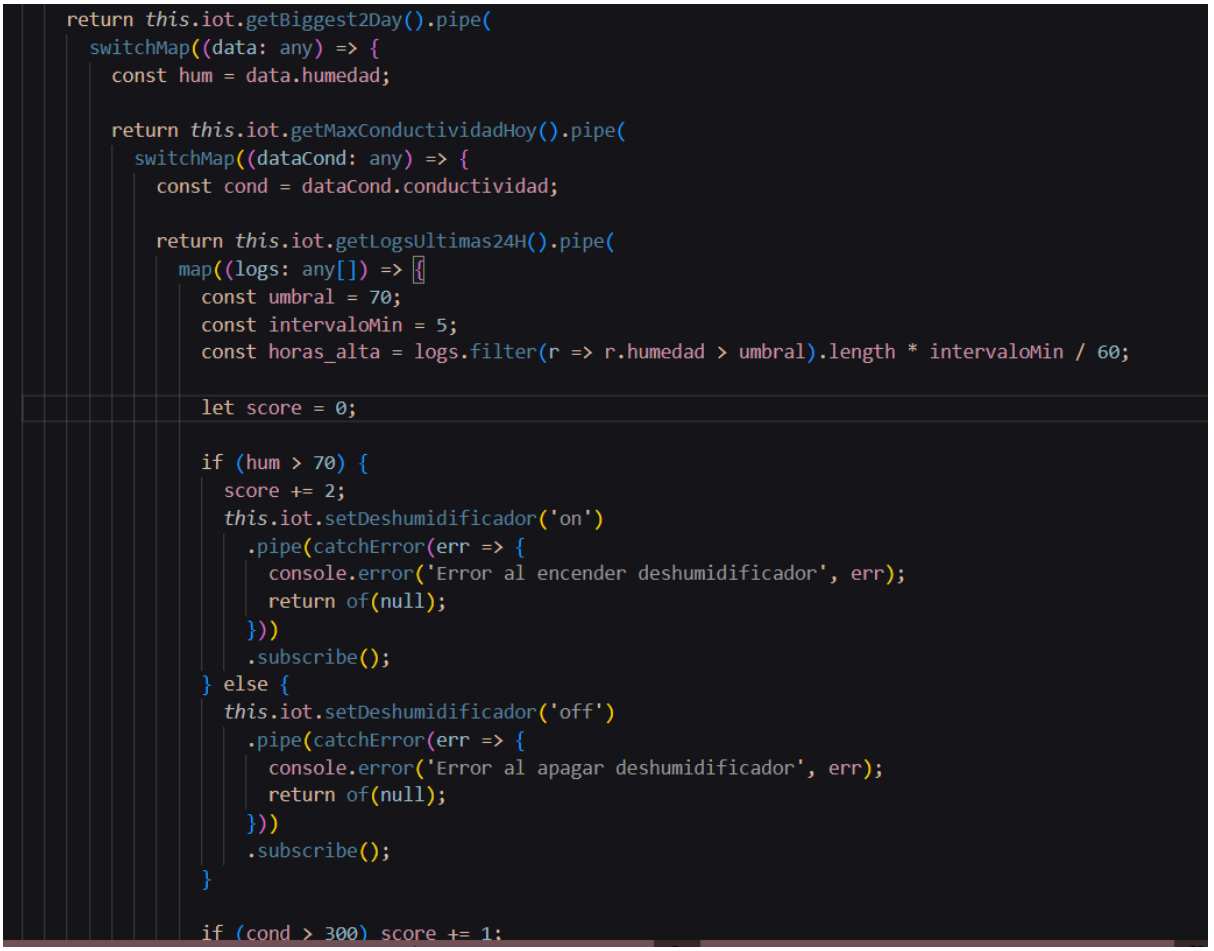
```python
# Control del deshumidificador
def desh_on():
    try:
        requests.post(f"{BASE_URL}/onDesH", timeout=3)
        estado["deshumidificador"] = "on"
    except Exception as e:
        print("Error encendiendo deshumidificador:", e)

def desh_off():
    try:
        requests.post(f"{BASE_URL}/offDesH", timeout=3)
        estado["deshumidificador"] = "off"
    except Exception as e:
        print("Error apagando deshumidificador:", e)
```

Figura 22. Código control del deshumidificador 1 (En cliente).

```
11    deshumidificador = false;
12    loading = false;
13
14    constructor(private iot: IotService) {}
15
16    toggleDeshumidificador() {
17      if (this.loading) return;
18      this.loading = true;
19
20      const accion = this.deshumidificador ? 'on' : 'off';
21
22      this.iot.setDeshumidificador(accion).subscribe({
23        next: (res) => {
24          console.log('Comando enviado correctamente:', res);
25          alert('Comando enviado correctamente');
26          this.loading = false;
27        },
28        error: (err) => {
29          console.error('Error al enviar comando:', err);
30          alert('No se pudo enviar el comando al deshumidificador');
31          this.deshumidificador = !this.deshumidificador;
32          this.loading = false;
33        }
34      });
35    }
36  }
```

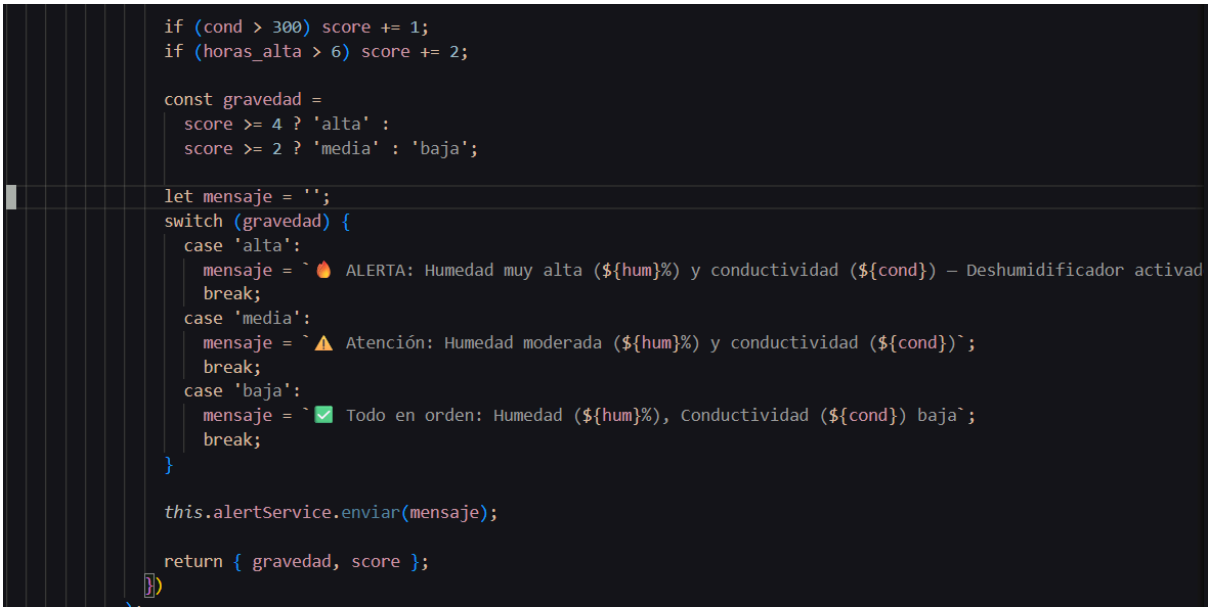
Figura 23. Código control del deshumidificador 2 (En Angular).

- **Módulo de procesamiento:** Implementa la lógica central del sistema, comparando los valores obtenidos con la tabla de niveles de gravedad. Incluye las funcionalidades de:
    - Monitoreo de humedad
    - Monitoreo de salinidad
    - Determinación del nivel de gravedad



```
return this.iot.getBiggest2Day().pipe(
  switchMap((data: any) => {
    const hum = data.humedad;

    return this.iot.getMaxConductividadHoy().pipe(
      switchMap((dataCond: any) => {
        const cond = dataCond.conductividad;

        return this.iot.getLogsUltimas24H().pipe(
          map((logs: any[]) => {
            const umbral = 70;
            const intervaloMin = 5;
            const horas_alta = logs.filter(r => r.humedad > umbral).length * intervaloMin / 60;

            let score = 0;

            if (hum > 70) {
              score += 2;
              this.iot.setDeshumidificador('on')
                .pipe(catchError(err => {
                  console.error('Error al encender deshumidificador', err);
                  return of(null);
                }))
                .subscribe();
            } else {
              this.iot.setDeshumidificador('off')
                .pipe(catchError(err => {
                  console.error('Error al apagar deshumidificador', err);
                  return of(null);
                }))
                .subscribe();
            }

            if (cond > 300) score += 1;
```

Figura 24. Código procesamiento 1.



```
    if (cond > 300) score += 1;
    if (horas_alta > 6) score += 2;

    const gravedad =
      score >= 4 ? 'alta' :
      score >= 2 ? 'media' : 'baja';

    let mensaje = '';
    switch (gravedad) {
      case 'alta':
        mensaje = `🔥 ALERTA: Humedad muy alta (${hum}%) y conductividad (${cond}) — Deshumidificador activad
        break;
      case 'media':
        mensaje = `⚠️ Atención: Humedad moderada (${hum}%) y conductividad (${cond})`;
        break;
      case 'baja':
        mensaje = `✅ Todo en orden: Humedad (${hum}%), Conductividad (${cond}) baja`;
        break;
    }

    this.alertService.enviar(mensaje);

    return { gravedad, score };
  })
```

Figura 25. Código procesamiento 2.

- **Módulo de recomendaciones:** Implementado para el caso de uso *"Sugerencias para plan de acción"*. Genera recomendaciones automáticas para el usuario en función del nivel de gravedad determinado por el sistema.

```
18    labelsUno: string[] = [
19      'Ventilar la casa diariamente al menos 15 minutos.',
20      'Usar deshumidificadores pequeños en habitaciones cerradas.',
21      'Evitar secar ropa dentro de la casa.'
22    ];
23
24    labelsDos: string[] = [
25      'Instalar extractores de aire en cocina y baño.',
26      'Colocar bolsas de gel de sílice en armarios.',
27      'Revisar filtraciones de agua en ventanas y techos.'
28    ];
29
30    labelsTres: string[] = [
31      'Usar deshumidificadores potentes con capacidad adecuada.',
32      'Reparar goteras y filtraciones inmediatamente.',
33      'Aplicar pintura antihumedad en paredes y techos.',
34      'Considerar revisión profesional de la estructura de la vivienda.'
35    ];
36
37    constructor(private gravedadService: GravedadService) {}
38
39    ngOnInit() {
40      // Evaluar la gravedad
41      this.gravedadService.evaluarHumedad().subscribe(resultado => {
42        this.gravedad = resultado.gravedad;
43        this.score = resultado.score;
44
45        // Seleccionar lista según gravedad
46        if (this.gravedad === 'baja') this.opcionSeleccionada = 'uno';
47        else if (this.gravedad === 'media') this.opcionSeleccionada = 'dos';
48        else if (this.gravedad === 'alta') this.opcionSeleccionada = 'tres';
49
```

Figura 26. Código recomendaciones.

- **Módulo de alertas:** Implementado para los casos de uso *"Aviso de nivel de gravedad"*. Permite notificar al usuario sobre eventos relevantes y condiciones físicas que requieren atención.

```
8   export class AlertaService {
9     private alertSource = new Subject<string>();
10    alert$ = this.alertSource.asObservable();
11
12    enviar(mensaje: string) {
13      this.alertSource.next(mensaje);
14    }
15  }
```

Figura 27. Código alertas (servicio angular).

```
export class AvisosComponent implements OnInit {
  mensajes: Aviso[] = [];
  maxAvisos = 10;

  constructor(private alertService: AlertaService) {}

  ngOnInit() {
    const guardados = localStorage.getItem('avisos');
    if (guardados) {
      const parsed = JSON.parse(guardados);
      this.mensajes = parsed.map((m: any) =>
        typeof m === 'string' ? { mensaje: m, hora: new Date().toLocaleTimeString() } : m
      );
    }

    this.alertService.alert$.subscribe(msg => {
      const nuevo: Aviso = { mensaje: msg, hora: new Date().toLocaleTimeString() };
      this.mensajes.push(nuevo);

      if (this.mensajes.length > this.maxAvisos) this.mensajes.shift();

      localStorage.setItem('avisos', JSON.stringify(this.mensajes));
    });
  }
}
```

Figura 28. Código alertas.

- **Módulo de interfaz de usuario:** Implementado mediante una aplicación web con diseño orientado a dispositivos móviles. Permite la visualización de datos, alertas, recomendaciones y el control del sistema de manera centralizada e intuitiva.

```
<div class="titulo-wrapper">
  <h1>Control</h1>
</div>
<div class="panel-dispositivos">
  <div class="dispositivo" [class.on]="false">
    <span class="icon">💨</span>
    <span class="nombre">Ventilador</span>
    <input type="checkbox" [checked]="false" disabled />
  </div>

  <div class="dispositivo" [class.on]="false">
    <span class="icon">💡</span>
    <span class="nombre">Luz</span>
    <input type="checkbox" [checked]="false" disabled />
  </div>

  <div class="dispositivo" [class.on]="deshumidificador">
    <span class="icon">💧</span>
    <span class="nombre">Deshumidificador</span>
    <input type="checkbox" [(ngModel)]="deshumidificador"
           [disabled]="loading"
           (change)="toggleDeshumidificador()" />
  </div>
</div>
```

Figura 29. Código interfaz (Ejemplo página de control html).

4.1.2.5 Reporte de Revisión

Durante la fase de implementación y pruebas unitarias, se realizó una revisión de código de prueba y funcionamiento. A continuación, se detallan las observaciones encontradas y las correcciones aplicadas para garantizar la estabilidad del sistema:

- Tiempo de diferencia entre cada recepción y entrega de datos entre microprocesador y periféricos.

4.2. Herramientas y técnicas

Todo proyecto, para su realización, necesita una variedad de herramientas para su respectivo desarrollo, en la que se pueden destacar las siguientes herramientas utilizadas:

Herramientas Implementadas:

- Python: Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software

Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo.

- Visual Studio Code: Visual Studio Code, que también se le conoce como VSCode, es un editor de código para programadores gratuito, de código abierto y multiplataforma. Está desarrollado por Microsoft, una compañía con una dilatada experiencia en la creación de IDEs (entornos de desarrollo integrados), que ha conseguido plasmar su larga tradición en el sector para ofrecer una herramienta ligera y práctica que la comunidad ha adoptado en masa.

- Raspberry Pi Os: Raspberry Pi OS (anteriormente llamado Raspbian) es una distribución del sistema operativo GNU/Linux basado en Debian, y por lo tanto libre para la SBC Raspberry Pi, orientado a la enseñanza de informática. El lanzamiento inicial fue en junio de 2012. Desde 2015, la Raspberry Pi Foundation lo ha proporcionado de forma oficial como el sistema operativo primario para la familia de placas SBC de Raspberry Pi.

- Redmine: Redmine es una herramienta para la gestión de proyectos, que con sus diversas funcionalidades permite a los usuarios de diferentes proyectos realizar el seguimiento y organización de los mismos. Además es posible optimizar su funcionamiento agregando funcionalidades. Incluye un sistema de seguimiento de incidentes con seguimiento de errores. Otras herramientas que incluye son calendario de actividades, diagramas de Gantt para la representación visual de la línea del tiempo de los proyectos, wiki, foro, visor del repositorio de control de versiones, RSS, control de flujo de trabajo basado en roles, integración con correo electrónico, entre otras opciones.

- Figma: Figma es un editor de gráficos vectorial y una herramienta de generación de prototipos, principalmente basada en la web, con características off-line adicionales habilitadas por aplicaciones de escritorio en macOS y Windows. Las aplicaciones Figma Mirror companion para Android y iOS permiten mirar los prototipos de Figma en dispositivos móviles. El conjunto de funciones de Figma, se enfoca en el uso de la interfaz de usuario y el diseño de experiencia de usuario, con énfasis en la colaboración en tiempo real.

Técnicas:

- Diseño de Interfaz de Usuario
- Prototipado de Experiencia
- Programación Orientada a Objetos (POO)
- Lectura y procesamiento de datos en tiempo real
- Uso de librerías especializadas para sensores
- Serialización y transmisión de datos
- Uso de peticiones HTTP
- Gestión de red y comunicación IoT
- Uso del entorno de desarrollo Linux
- Control de hardware mediante grovepi
- Configuración de comunicación entre microcontrolador y microprocesador
- Carta Gantt

- Gestión de Riesgos

5 Planificación de procesos de soporte

5.1 Planificación de la documentación

● Manual de usuario: El objetivo de este manual es que una persona sin conocimientos técnicos pueda instalar, configurar y usar el sistema de monitoreo y control de humedad de manera efectiva. Debe ser una guía práctica y fácil de seguir.

● Wiki de Redmine: Una Wiki es un repositorio central de información técnica y de gestión para el equipo de desarrollo, que permite organizar y actualizar la documentación de manera colaborativa.

● Documentación del código: Es el material directamente integrado en el código fuente y los documentos generados a partir de él, que explican el funcionamiento interno de cada módulo.

6. Problemas encontrados

- **Problema 1: Retraso en obtención de materiales**
  Hubo un retraso significativo debido a la falta de sensores y actuadores adecuados disponibles en la ciudad.

- **Problema 2: Errores de código**
  Se encontraron errores en el código respecto al traspaso de datos con backend y el uso de API KEY para open al, que no se han logrado corregir del todo debido a la falta de disponibilidad del hardware y tiempo

- **Problema 3: Error de compatibilidad y reconocimiento de paquetes APK en Meta Quest**
  Al intentar realizar el despliegue de la maqueta virtual desde Unity hacia los visores Meta Quest 3, el dispositivo no reconocía el archivo APK generado. Este error de lectura impidió la visualización del modelo 3D en el entorno de realidad virtual, provocando un retraso significativo en las pruebas de usabilidad y en el cronograma establecido para la validación de la maqueta.

7. Soluciones propuestas

- **Solución al problema 1:** Solicitar al departamento el préstamo de componentes equivalentes de otros kits o reasignar el presupuesto de $20.000 para compras locales de emergencia, priorizando los sensores de humedad y conductividad que son el núcleo del sistema.
- **Solución al problema 2:** Implementar un sistema de manejo de excepciones en Python que permita al sistema seguir operando de forma básica (monitoreo local)

aunque falle la conexión con el backend o la API de OpenAI, además de documentar los errores en la Wiki de Redmine para soporte colaborativo.
- **Solución al problema 3:** Se realizó una reconfiguración completa de los Build Settings en Unity, asegurando el cambio de plataforma a Android y la instalación de los paquetes necesarios.

8. Trabajo a futuro

Se han identificado las siguientes ideas para la mejora y expansión de este proyecto:

- Integración de Inteligencia Artificial (IA): Implementar modelos de aprendizaje automático para predecir cuándo ocurrirán picos de humedad basándose en datos históricos del clima de Arica.

- Expansión a Ambientes Industriales: Adaptar el sistema para que pueda ser utilizado en bodegas o almacenes a gran escala, superando el alcance experimental actual.

- Optimización del Consumo Energético: Desarrollar un modo de "bajo consumo" para el ESP8266 y la Raspberry Pi 4B, permitiendo que el sistema funcione con paneles solares en zonas con poca conectividad eléctrica.

- Conectividad en la Nube (Cloud): Migrar la base de datos local a una plataforma en la nube (como AWS o Firebase) para permitir el acceso a los niveles de salinidad desde cualquier lugar del mundo, eliminando la restricción del Wi-Fi local.

- Módulo de Autolimpieza: Diseñar un mecanismo que limpie automáticamente el sensor de conductividad y el contenedor del deshumidificador para evitar lecturas erróneas por acumulación de residuos.

- Notificaciones mediante Asistentes de Voz: Integrar el sistema con Alexa o Google Home para que el usuario reciba alertas de gravedad por voz en tiempo real.

9. Conclusión

En conclusión, este proyecto consolidó la planificación de una infraestructura tecnológica diseñada específicamente para enfrentar la degradación de viviendas causada por factores ambientales en zonas costeras. Bajo el nombre de “Sistema de Monitoreo y Regulación de Humedad”, el trabajo integra el uso de hardware de código abierto (Raspberry Pi 4B y ESP8266) para articular una respuesta automatizada frente al moho y la salinidad, dos amenazas constantes para la salud y la edificación en la región de Arica.

La solución se apoya en un ecosistema digital donde el usuario no es solo un espectador, sino un administrador activo a través de una aplicación web. En esta plataforma, se implementaron controles para gestionar dispositivos de mitigación como la luz UV y ventiladores, además de visualizar diagnósticos de gravedad basados en datos reales capturados por sensores de alta precisión.

El éxito de este diseño se basó en una fase de análisis crítico donde se estudió el fenómeno de la camanchaca y su impacto en los materiales de construcción. Esta investigación permitió establecer un conjunto de requisitos funcionales y no funcionales sumamente detallados, garantizando que el sistema sea capaz de reaccionar en menos de 2 segundos ante cualquier anomalía detectada.

Finalmente, la arquitectura propuesta no solo soluciona la problemática actual, sino que proyecta una estructura escalable y de fácil mantenimiento gracias a la separación de sus módulos lógicos y físicos. A pesar de los desafíos logísticos con el hardware y las pruebas en realidad virtual, se entrega una base sólida que prioriza la seguridad habitacional y el bienestar ambiental, cumpliendo con la misión fundamental de prevenir daños antes de que estos se vuelvan irreparables

10. Referencias

Weatherspark. (s. f.). *Clima promedio en Arica, Chile durante todo el año.* Recuperado el 28 de octubre de 2025, de https://es.weatherspark.com/y/26550/Clima-promedio-en-Arica-Chile-durante-todo-el-a%C3%B1o

Pérez, M. A., & Díaz, L. F. (2023). *[EVALUACIÓN DE LA HUMEDAD POR CONDENSACIÓN DENTRO DE VIVIENDAS SOCIALES]*. *Revista INVI, 38*(107), 1–28. Universidad de Chile. Recuperado de https://revistainvi.uchile.cl/index.php/INVI/article/view/62167/66277

eBay. (s. f.). *GrovePi+ Starter Kit* [Anuncio en eBay]. Recuperado el 28 de octubre de 2025, de https://www.ebay.com/itm/404401501321?_skw=GrovePi%2B+Kit+de+inicio&itmmeta=01K8NJ61NVX6W8KXV8ZXF6HZ6B&hash=item5e28353489:g:gcYAAOSwhH9kv95t&itmprp=enc%3AAQAKAAAA0FkggFvd1GGDu0w3yXCmi1dkuuxTXohGB6SSo1Wuqz0lTImmB4ExSHogq%2Flu06NKCh4UEinOjsRSE9z335lQrlmF852l0X8vQ30TtX4sSinliYGBBr%2FvvZn%2Bz6xf1xm9oHwiEnfuoJ0gqLDdFLRX%2FLJuRO6rqdOBQrWax59tOsBQ6UPKq5jCEUR7Cgld7dUv7WGfBs3AU3x3AnzJ2BidrRtdjP2VEYkw4zg6AFOH2SLMVsNT%2BQGMUUHL3y30AKP8g01S8%2BkeaEzczXtySLZvo0c%3D%7Ctkp%3ABk9SR4KbmLLFZg

Falabella. (s. f.). *Thermalright Ventilador Negro TL-C12C.* Recuperado el 28 de octubre de 2025, de https://www.falabella.com/falabella-cl/product/146084436/Thermalright-Ventilador-Negro-TL-C12C/146084437

AliExpress. (s. f.). *Ventilador Tesla (modelo no especificado).* Recuperado el 28 de octubre de 2025, de https://www.aliexpress.com/p/tesla-landing/index.html

Paris. (s. f.). *Kit Raspberry Pi 4 8 GB RAM con carcasa oficial*. Recuperado el 28 de octubre de 2025, de https://www.paris.cl/kit-raspberry-pi-4-8gb-ram-con-carcasa-oficial-MK2JAS7TSD.html

AliExpress. (s. f.). *[Sensor de conductividad del agua]*. Recuperado el 28 de octubre de 2025, de https://www.aliexpress.com/p/tesla-landing/index.html?scenario=c_ppc_item_bridge&productId=1005001698342637…

Mercado Libre Chile. (s. f.). *Lámpara UV de ozono de 8 W – lámpara germicida UV.* Recuperado el 28 de octubre de 2025, de https://www.mercadolibre.cl/w-lampara-uv-de-ozono-de-8-w-lampara-germicida-uv/p/MLC2030083045?utm_source

Falabella. (s. f.). *ESP WiFi Bluetooth DevKit v1 de 30 pines compatible con Arduino.* Recuperado el 28 de octubre de 2025, de https://www.falabella.com/falabella-cl/product/130037662/ESP-Wifi-Bluetooth-Devkit-v1-30-pines-compatible-con-Arduino/130037663

Paris. (s. f.). *Notebook Pavilion Plus 14 eh0101la – Intel Core i7, 16 GB RAM, 512 GB SSD, NVIDIA GeForce RTX 2050.* Recuperado el 28 de octubre de 2025, de https://www.paris.cl/notebook-pavilion-plus-14-eh0101la-intel-core-i7-16gb-ram-512gb-ssd-nvidia-geforce-rtx-2050-14-28k-139160999.html

Cybertech. (s. f.). *MSI Notebook Gamer Katana GF66 11SC de 15,6".* Recuperado el 28 de octubre de 2025, de https://www.cybertech.cl/msi-notebook-gamer-katana-gf66-11sc-de-156-1

Paris. (s. f.). *Lenovo ThinkPad X390 Yoga, i7-8565U, 16 GB RAM, 512 GB SSD (reacondicionado).* Recuperado el 28 de octubre de 2025, de https://www.paris.cl/lenovo-thinkpad-x390-yoga-i7-8565u-16gb-ram-512gb-ssd-133-w10p-reacondicionado-MKG772878L.html