

# **SISTEMA PARA LA DETECCIÓN DE ESTRÉS HÍDRICO EN EL ARÁNDANO BILOXI MEDIANTE TERMOGRAFÍA DE BAJO COSTO**

## **AUTOR(ES)**

Juan Esteban Fuentes Rojas

Gabriel Esteban Martinez Roldan

**UNIVERSIDAD DE CUNDINAMARCA**

Facultad de Ingeniería

Programa de Ingeniería de Sistemas y Computación

Facatativá, Noviembre 2025

# **SISTEMA PARA LA DETECCIÓN DE ESTRÉS HÍDRICO EN EL ARÁNDANO BILOXI MEDIANTE TERMOGRAFÍA DE BAJO COSTO**

## **AUTOR(ES)**

Directora: Ing. Gina Maribel Valenzuela Sabogal  
Juan Esteban Fuentes Rojas  
Gabriel Esteban Martinez Roldan

**GRUPO DE INVESTIGACIÓN DE SISTEMAS Y TECNOLOGÍA DE FACATATIVÁ  
(GISTFA)**

**UNIVERSIDAD DE CUNDINAMARCA**  
Facultad de Ingeniería  
Programa de Ingeniería de Sistemas  
Facatativá, Noviembre 2025

# **Dedicatoria**

Texto de la dedicatoria...

# **Agradecimientos**

Texto de los agradecimientos...

# Resumen

La producción de arándano, especialmente de la variedad Biloxi, está en expansión en el altiplano cundiboyacense (Quintana, 2020). Sin embargo, este cultivo presenta una alta sensibilidad a la disponibilidad de agua debido a su sistema radicular superficial, lo que lo hace vulnerable al estrés hídrico y puede comprometer significativamente el rendimiento y la calidad de la fruta (Morales, 2017). La termografía infrarroja (IRT) es una técnica no invasiva que permite monitorear el estado fisiológico de las plantas. Su principio se basa en que el estrés hídrico provoca un cierre estomático que reduce la transpiración, causando un aumento en la temperatura de la superficie foliar detectable con cámaras térmicas (García Tejero et al., 2015). A pesar de su potencial, el alto costo de los equipos comerciales ha limitado su adopción generalizada. Este proyecto de investigación propone el desarrollo de un sistema de bajo costo para la detección temprana de estrés hídrico en el arándano Biloxi. El objetivo es diseñar y construir una herramienta que capture y procese datos termográficos, permitiendo a los agricultores tomar decisiones informadas sobre el manejo del riego. Con ello se busca optimizar el uso del agua, minimizar las pérdidas económicas asociadas al estrés hídrico y fomentar prácticas agrícolas más sostenibles. Para el desarrollo del proyecto se empleará una metodología de investigación mixta, combinando el análisis cuantitativo de los datos termográficos con la observación cualitativa del estado de las plantas. Adicionalmente, la construcción del sistema se gestionará bajo el marco de trabajo ágil Scrum, lo que permitirá un desarrollo iterativo y adaptable a las necesidades del proyecto.

**Palabras clave:** Termografía, Estrés Hídrico, Arándano Biloxi

# Abstract

Blueberry production, especially of the Biloxi variety, is expanding in the Cundiboyacense highlands (Quintana, 2020). However, this crop is highly sensitive to water availability due to its shallow root system, which makes it vulnerable to water stress and can significantly compromise fruit yield and quality (Morale2017). Infrared thermography (IRT) is a non-invasive technique that allows the physiological state of plants to be monitored. Its principle is based on the fact that water stress causes stomatal closure, which reduces transpiration, causing an increase in leaf surface temperature that can be detected with thermal cameras (García Tejero et al., 2015). Despite its potential, the high cost of commercial equipment has limited its widespread adoption. This research project proposes the development of a low-cost system for the early detection of water stress in Biloxi blueberries. The objective is to design and build a tool that captures and processes thermographic data, allowing farmers to make informed decisions about irrigation management. The aim is to optimize water use, minimize economic losses associated with water stress, and promote more sustainable agricultural practices. A mixed research methodology will be used to develop the project, combining quantitative analysis of thermographic data with qualitative observation of plant condition. In addition, the construction of the system will be managed under the agile Scrum framework, which will allow for iterative development that can be adapted to the needs of the project.

**Keywords:** Water stress, Biloxi blueberry, Thermography.

# Índice general

<b>Dedicatoria</b>	<b>II</b>
<b>Agradecimientos</b>	<b>III</b>
<b>Resumen</b>	<b>IV</b>
<b>Abstract</b>	<b>V</b>
<b>Lista de Tablas</b>	<b>IX</b>
<b>Lista de Figuras</b>	<b>XII</b>
<b>Lista de Anexos</b>	<b>XIII</b>
<b>Introducción</b>	<b>XIV</b>
<b>1 INFORME DE INVESTIGACIÓN</b>	<b>1</b>
1.1 Estado del Arte . . . . .	1
1.2 Línea de Investigación . . . . .	1
1.3 Planteamiento del Problema y Pregunta de Investigación . . . . .	1
1.4 Objetivo General y Objetivos Específicos . . . . .	3
1.4.1 Objetivo General . . . . .	3
1.4.2 Objetivos Específicos . . . . .	3
1.5 Alcance e Impacto del Proyecto . . . . .	3
1.6 Metodología . . . . .	4
1.6.1 Metodología de Investigación . . . . .	4
1.6.2 Metodología de Desarrollo . . . . .	5
1.7 Marcos de Referencia . . . . .	6
1.7.1 Marco Teórico . . . . .	6
1.7.2 Marco Legal . . . . .	6
<b>2 DOCUMENTACIÓN SOFTWARE</b>	<b>9</b>
2.1 Plan de Proyecto . . . . .	9
2.2 Arquitectura del Software . . . . .	11
2.2.1 Patrón Arquitectónico y Diseño Lógico . . . . .	12

2.2.2	Arquitectura Física e Infraestructura . . . . .	14
2.2.3	Stack Tecnológico . . . . .	16
2.3	Determinación de Requerimientos . . . . .	19
2.4	Especificación del Diseño . . . . .	25
2.4.1	Modelo de Entidad-Relación (MER) . . . . .	25
2.4.2	Diagramas de Casos de Uso . . . . .	29
2.4.3	Diagramas de Secuencia . . . . .	44
2.4.4	Diagramas de Actividades . . . . .	76
2.4.5	Diagrama de Clases . . . . .	107
2.4.6	Diagrama de Despliegue . . . . .	121
2.5	Diseño de los Casos de Prueba . . . . .	123
2.6	Estimación de Recursos . . . . .	123
2.7	Resultados de la Implementación del Software . . . . .	123
2.8	Conclusiones y Recomendaciones del software . . . . .	124
<b>3</b>	<b>DOCUMENTACIÓN HARDWARE</b>	<b>125</b>
3.1	Introducción . . . . .	125
3.2	Objetivos . . . . .	125
3.3	Descripción de Componentes . . . . .	125
3.3.1	Microcontrolador ESP32-S3-WROOM-1 N16R8 . . . . .	125
3.3.2	Cámara termográfica MLX90640 . . . . .	126
3.3.3	Sensor de luz BH1750 . . . . .	126
3.3.4	Sensor de humedad y temperatura DHT22 . . . . .	126
3.3.5	Cámara RGB OV2640 . . . . .	126
3.3.6	Regulador de voltaje LM2596 . . . . .	126
3.4	Metodología de Caracterización . . . . .	126
3.4.1	Evaluación y verificación de componentes . . . . .	126
3.4.2	Configuración e Integración del Firmware . . . . .	127
3.4.3	Validación y Análisis de Resultados . . . . .	127
3.5	Implementación del Sistema Integrado . . . . .	128
3.6	Resultados . . . . .	128
<b>4</b>	<b>ESTUDIO EXPERIMENTAL</b>	<b>129</b>
4.1	Introducción . . . . .	129
4.2	Objetivos del Estudio Experimental . . . . .	130

4.2.1	Objetivo General . . . . .	130
4.2.2	Objetivos Específicos . . . . .	130
4.3	Materiales y Métodos . . . . .	131
4.3.1	Arquitectura del Hardware . . . . .	131
4.3.2	Validación Metrológica del Sensor Térmico . . . . .	132
4.4	Diseño Experimental . . . . .	132
4.5	Recolección y Análisis de Datos . . . . .	133
4.5.1	Recolección de Datos . . . . .	133
4.5.2	Procesamiento y Análisis de Datos . . . . .	134
4.6	Resultados . . . . .	135
<b>5</b>	<b>Resultados y Conclusiones Finales</b>	<b>137</b>
5.1	Respuesta a la Pregunta de Investigación . . . . .	137
5.2	Discusión General . . . . .	138
5.2.1	Consecución de los Objetivos Planteados . . . . .	138
5.2.2	Fortalezas y Limitaciones del Sistema . . . . .	139
5.2.3	Impacto Potencial . . . . .	140
5.3	Recomendaciones y Trabajo Futuro . . . . .	141
5.4	Conclusiones Finales . . . . .	142
<b>Bibliografía</b>		<b>143</b>
Referencias . . . . .		143

# **Lista de Tablas**

1	<i>Requerimiento Funcional RF01: Registro</i>	19
2	<i>Requerimiento Funcional RF02: Inicio de sesión</i>	19
3	<i>Requerimiento Funcional RF03: CRUD cámara</i>	20
4	<i>Requerimiento Funcional RF04: CRUD persona</i>	20
5	<i>Requerimiento Funcional RF05: Módulo de mediciones</i>	21
6	<i>Requerimiento Funcional RF06: Módulo de procesamiento</i>	21
7	<i>Requerimiento Funcional RF07: Reportes</i>	22
8	<i>Requerimiento Funcional RF08: Notificaciones</i>	22
9	<i>Requerimiento Funcional RF09: Gestionar Observaciones</i>	23
10	<i>Requerimiento No Funcional RNF01: Seguridad</i>	23
11	<i>Requerimiento No Funcional RNF02: Copia de seguridad</i>	24

# **Lista de Figuras**

1	Diagrama de Gantt del proyecto. . . . .	10
2	Diagrama de Despliegue Físico del Sistema Arandano IRT. . . . .	15
3	Diagrama Entidad-Relación del Sistema. . . . .	25
4	Diagrama de Casos de Uso para la Gestión de Usuarios (RF1, RF2). . . . .	30
5	Diagrama de Casos de Uso para la Gestión de Cámaras (RF3). . . . .	32
6	Diagrama de Casos de Uso para la Gestión de Perfiles (RF4). . . . .	34
7	Diagrama de Casos de Uso para el Módulo de Mediciones (RF5). . . . .	36
8	Diagrama de Casos de Uso para la Gestión de Plantas (RF6). . . . .	38
9	Diagrama de Casos de Uso para la Generación de Reportes (RF7). . . . .	40
10	Diagrama de Casos de Uso para las Notificaciones (RF8). . . . .	42
11	Diagrama de Casos de Uso para el Módulo de Observaciones (RF9). . . . .	43
12	Diagrama de secuencia base: Envío de formulario. . . . .	47
13	Diagrama de Secuencia para el Registro (RF1.0). . . . .	49
14	Diagrama de Secuencia para Solicitar Código (RF1.1). . . . .	50
15	Diagrama de Secuencia para Iniciar Sesión (RF2.0). . . . .	51
16	Diagrama de Secuencia para Cerrar Sesión (RF2.1). . . . .	52
17	Diagrama de Secuencia para Recuperar Contraseña (RF2.2). . . . .	53
18	Diagrama de Secuencia para Crear Cámara (RF3.1). . . . .	54
19	Diagrama de Secuencia para Activar Cámara (RF3.1.1). . . . .	55
20	Diagrama de Secuencia para Consultar Cámara (RF3.2). . . . .	57
21	Diagrama de Secuencia para Editar Cámara (RF3.3). . . . .	58
22	Diagrama de Secuencia para Eliminar Cámara (RF3.4). . . . .	59
23	Diagrama de Secuencia para Consultar Perfil (RF4.1). . . . .	60
24	Diagrama de Secuencia para Editar Perfil (RF4.2). . . . .	61
25	Diagrama de Secuencia para Eliminar Perfil (RF4.3). . . . .	62
26	Diagrama de Secuencia para Cambiar Contraseña (RF4.4). . . . .	63

27	Diagrama de Secuencia para Agregar Integrante de Cultivo (RF4.5). . . . .	64
28	Diagrama de Secuencia para Eliminar Integrante de Cultivo (RF4.6). . . . .	65
29	Diagrama de Secuencia para el Módulo de Mediciones (RF5.0). . . . .	66
30	Diagrama de Secuencia para Crear Planta (RF6.1). . . . .	67
31	Diagrama de Secuencia para Consultar Planta (RF6.2). . . . .	68
32	Diagrama de Secuencia para Editar Planta (RF6.3). . . . .	69
33	Diagrama de Secuencia para Eliminar Planta (RF6.4). . . . .	70
34	Diagrama de Secuencia para Generar Reporte (RF7.0). . . . .	71
35	Diagrama de Secuencia para Descargar Reporte (RF7.1). . . . .	71
36	Diagrama de Secuencia para Adjuntar Reporte (RF7.2). . . . .	72
37	Diagrama de Secuencia para Notificar Planta (RF8.1). . . . .	73
38	Diagrama de Secuencia para Notificar Seguridad (RF8.2). . . . .	74
39	Diagrama de Secuencia para Crear Observación (RF9.1). . . . .	75
40	Diagrama de Secuencia para Consultar Observación (RF9.1). . . . .	75
41	Diagrama de Actividad para el Registro (RF1.0). . . . .	78
42	Diagrama de Actividad para Solicitar Código (RF1.1). . . . .	79
43	Diagrama de Actividad para Iniciar Sesión (RF2.0). . . . .	80
44	Diagrama de Actividad para Cerrar Sesión (RF2.1). . . . .	81
45	Diagrama de Actividad para Recuperar Contraseña (RF2.2). . . . .	82
46	Diagrama de Actividad para Crear Cámara (RF3.1). . . . .	83
47	Diagrama de Actividad para Activar Cámara (RF3.1.1). . . . .	84
48	Diagrama de Actividad para Consultar Cámara (RF3.2). . . . .	86
49	Diagrama de Actividad para Editar Cámara (RF3.3). . . . .	87
50	Diagrama de Actividad para Eliminar Cámara (RF3.4). . . . .	88
51	Diagrama de Actividad para Consultar Perfil (RF4.1). . . . .	89
52	Diagrama de Actividad para Editar Perfil (RF4.2). . . . .	90
53	Diagrama de Actividad para Eliminar Perfil (RF4.3). . . . .	91
54	Diagrama de Actividad para Cambiar Contraseña (RF4.4). . . . .	92
55	Diagrama de Actividad para Agregar Integrante de Cultivo (RF4.5). . . . .	93
56	Diagrama de Actividad para Eliminar Integrante de Cultivo (RF4.6). . . . .	94
57	Diagrama de Actividad para el Módulo de Mediciones (RF5.0). . . . .	95
58	Diagrama de Actividad para Crear Planta (RF6.1). . . . .	96
59	Diagrama de Actividad para Consultar Planta (RF6.2). . . . .	97

60	Diagrama de Actividad para Editar Planta (RF6.3). . . . .	98
61	Diagrama de Actividad para Eliminar Planta (RF6.4). . . . .	99
62	Diagrama de Actividad para Generar Reporte (RF7.0). . . . .	100
63	Diagrama de Actividad para Descargar Reporte (RF7.1). . . . .	101
64	Diagrama de Actividad para Adjuntar Reporte (RF7.2). . . . .	102
65	Diagrama de Actividad para Notificar Planta (RF8.1). . . . .	103
66	Diagrama de Actividad para Notificar Seguridad (RF8.2). . . . .	104
67	Diagrama de Actividad para Crear Observación (RF9.1). . . . .	105
68	Diagrama de Actividad para Consultar Observación (RF9.2). . . . .	106
69	Diagrama de Clases: Entidades Principales y Datos de Monitoreo. . . . .	107
70	Diagrama de Clases: Gestión de Usuarios y Roles. . . . .	109
71	Diagrama de Clases: Ciclo de Vida y Seguridad del Dispositivo. . . . .	111
72	Diagrama de Clases: Configuraciones de Cultivo. . . . .	112
73	Diagrama de Clases: Gestión de Usuarios y Autenticación (Capa de Aplicación). .	113
74	Diagrama de Clases: Gestión de Entidades (CRUD) (Capa de Aplicación). . . . .	115
75	Diagrama de Clases: Interacción con Dispositivos y Envío de Datos (Capa de Aplicación). . . . .	116
76	Diagrama de Clases: Arquitectura de Peticiones y Seguridad. . . . .	118
77	Diagrama de Clases: Patrón de Controladores y Vistas (MVC / API). . . . .	120
78	Diagrama de Despliegue del Sistema. . . . .	122
79	Diagrama del prototipo de hardware utilizado en el experimento. . . . .	132

# **Lista de Anexos**

1. Anexo A: Título del Anexo A
2. Anexo B: Título del Anexo B

# **Introducción**

Texto de la introducción (2 a 4 páginas).

# **I. INFORME DE INVESTIGACIÓN**

## **1.1. Estado del Arte**

Texto del estado del arte...

## **1.2. Línea de Investigación**

Texto de la línea de investigación...

## **1.3. Planteamiento del Problema y Pregunta de Investigación**

La producción de arándanos está en aumento debido a su alta demanda, siendo la variedad Biloxi la que más se cultiva en el altiplano cundiboyacense según datos de Quintana (2020). Sin embargo, la planta requiere condiciones específicas de temperatura para dar fruto. La falta de estas condiciones la hace susceptible a enfermedades, representando un desafío significativo para los productores. Sin un manejo oportuno, las enfermedades pueden causar la muerte de las plantas, afectando el cultivo y provocando pérdidas económicas. Esto subraya la importancia de una gestión adecuada, reflejada en las exportaciones de frutos del género Vaccinium en Colombia, que alcanzaron los 2,2 millones de dólares, según la Asociación Nacional de Comercio Exterior (ANALDEX, 2022).

Respecto a las enfermedades en plantas (fitopatologías), Quintana afirma que la *Botrytis Cinerea* es una de las más comunes en el arándano. La *Botryotinia Fuckeliana* (fase asexuada: *Botrytis Cinerea*), comúnmente conocida como *Botrytis* o moho gris, es una enfermedad fúngica que afecta una amplia variedad de plantas, incluyendo la planta de arándano Biloxi (Quintana, 2020). Esta enfermedad es particularmente destructiva en condiciones de alta humedad y temperaturas moderadas, promoviendo la formación de esporas. Los síntomas típicos de la *Botrytis* incluyen manchas marrones en hojas, flores y frutos, que eventualmente se cubren de un moho gris característico. En las hojas, causa lesiones de color café que comienzan generalmente por el centro de la lámina y se extienden hacia los bordes, produciendo una necrosis extensiva. En condiciones de alta humedad, sobre las lesiones de las hojas se desarrollan las estructuras reproductivas del patógeno (conidióforos y conidios), que dan un aspecto plomizo (grisáceo o plateado) en los tejidos (Morales, et al. 2017).

Una técnica no invasiva ampliamente utilizada en agricultura para monitorear la salud de las plantas es la termografía infrarroja (IRT), permitiendo además gestionar el riego, detectar enfermedades y estimar la producción (Dong et al., 2024; Aux et al., 2022). Esta herramienta es fundamental para avanzar hacia una agricultura más automatizada, precisa y sostenible, permitiendo supervisar el estrés térmico en cultivos y analizar el impacto de patógenos en la transpiración de las plantas. Aunque la investigación demuestra que la IRT está superando sus limitaciones y se está convirtiendo en un método robusto, confiable y económico para determinar el estado hídrico de las plantas y detectar el estrés (Pineda et al., 2020), su alto costo actual restringe su uso principalmente a grupos de investigación y empresas especializadas (García Tejero et al., 2015).

Teniendo en cuenta las afectaciones de la *Botrytis* a los cultivos de arándano, es crucial explorar métodos no invasivos como la termografía. Si la termografía permite hacer una detección temprana de esta enfermedad se podría mejorar la gestión de la salud del cultivo y reducir las pérdidas. Lo cual plantea la pregunta, ¿Cómo desarrollar un sistema para aproximar la detección temprana de

Botrytis Cinerea en plantas de arándanos Biloxi mediante hardware de bajo costo?

## **1.4. Objetivo General y Objetivos Específicos**

### **1.4.1. Objetivo General**

Desarrollar un sistema basado en termografía infrarroja (IRT) y hardware de bajo costo para la detección del estrés hídrico en plantas de arándano Biloxi.

### **1.4.2. Objetivos Específicos**

1. Identificar y documentar los requisitos funcionales y no funcionales del sistema.
2. Modelar la arquitectura del sistema mediante la creación de diagramas UML.
3. Integrar el hardware del módulo termográfico para la recolección de datos.
4. Desarrollar el software que recolecte los datos del módulo termográfico para su posterior procesamiento.
5. Evaluar la generación de un reporte de estado hídrico que diferencie entre plantas de arándano Biloxi con riego óptimo y con déficit, a partir de los datos recopilados por el sistema.

## **1.5. Alcance e Impacto del Proyecto**

El presente proyecto busca ser un apoyo a la producción de arándanos a través de la implementación de tecnologías de precisión, como la termografía, para monitorear y gestionar la salud de los cultivos. Esto podría permitir a los agricultores minimizar las perdidas en los cultivos y poder “satisfacer los desafíos de seguridad alimentaria local, regional y global del siglo XXI” (Vargas Q. & Best

S., 2021). La agricultura de precisión, apoyada en tecnologías como la termografía, es fundamental para una producción más sostenible, ya que permite optimizar el uso de recursos críticos como el agua (Pineda et al., 2021). La capacidad de monitorear el estado hídrico de las plantas en tiempo real posibilita una gestión del riego más eficiente, aplicando agua solo cuando y donde es realmente necesario (García Tejero et al., 2015). Este enfoque no solo promueve la conservación del recurso hídrico, sino que también contribuye a que los agricultores sean más resilientes y competitivos (Dong et al., 2024). Este proyecto está alineado con los Objetivos de Desarrollo Sostenible (ODS), específicamente con el ODS 9, que busca promover la inversión en infraestructura y la innovación para impulsar el crecimiento económico y el desarrollo sostenible. También contribuye a reducir la huella ecológica mediante la gestión eficiente de los recursos naturales compartidos y la implementación de prácticas agrícolas sostenibles, lo que se relaciona con el ODS 12. La implementación de este proyecto puede tener un impacto significativo en la seguridad alimentaria y la economía local. Al reducir las pérdidas económicas en la producción de cultivos y al promover la eficiencia energética, podemos crear cadenas de producción y suministro más eficientes. Además, el acceso a tecnologías de precisión puede ayudar a reducir la brecha digital entre países desarrollados y en desarrollo, lo que se relaciona con el ODS 9.

## **1.6. Metodología**

### **1.6.1. Metodología de Investigación**

Para llevar a cabo esta investigación se utilizará la metodología de investigación mixta, la cual combina la perspectiva cuantitativa y cualitativa, lo que permite dar profundidad al análisis y comprender mejor los procesos. Esto implica recopilar, analizar e interpretar datos tanto cualitativos

como cuantitativos para obtener una visión más completa de la problemática a estudiar. Esta metodología busca compensar las limitaciones de cada enfoque al mismo tiempo que fortalece la validez de la interpretación de los resultados (Hamui-Sutton, 2013).

La metodología mixta en este proyecto se empleará con el fin de combinar la recopilación y análisis de datos cuantitativos, como las lecturas de temperatura obtenidas a través del módulo de cámara térmica, con la exploración cualitativa de los resultados, como la descripción de las características en las plantas de arándano biloxi. Además de los datos termográficos, también se tomarán en cuenta otras variables importantes para el crecimiento de la planta y el hongo, buscando así tener una visión más completa de los factores que influyen en su desarrollo. Esto permitirá obtener una vista más amplia de la problemática a estudiar, comprendiendo así la relación entre las lecturas termográficas y la presencia de *Botrytis Cinerea*, además de aquellos factores que puedan influir en la precisión del sistema.

### **1.6.2. Metodología de Desarrollo**

Para el desarrollo del sistema, se adoptarán elementos del marco de trabajo ágil Scrum, un enfoque de desarrollo iterativo y colaborativo diseñado para fomentar la adaptabilidad y la entrega continua de valor. Este método, reconocido por su flexibilidad y eficiencia, se basa en ciclos cortos de trabajo llamados Sprints, que permiten planificar, ejecutar y revisar tareas de manera organizada y dinámica (SCRUMstudy™, 2022).

Durante el proyecto, se trabajará en Sprints semanales, en los cuales se evaluarán los avances del Product Backlog previamente definido. Esta estructura facilita la retroalimentación constante, priorización de tareas y ajustes según las necesidades del proyecto, asegurando un desarrollo estructurado y orientado a cumplir los objetivos establecidos. La implementación de Scrum permitirá un desarrollo funcional que cumpla con las expectativas de precisión y eficacia requeridas.

## **1.7. Marcos de Referencia**

### **1.7.1. Marco Teórico**

Texto del marco teórico...

### **1.7.2. Marco Legal**

El desarrollo del presente proyecto de investigación y del sistema tecnológico asociado se enmarca en la normativa colombiana vigente, abordando tres ejes fundamentales: la propiedad intelectual, la protección de datos personales y la seguridad digital.

#### **Propiedad Intelectual y Derechos de Autor**

La titularidad y gestión de los derechos de autor generados en este proyecto se rigen por el **Acuerdo No. 04 de 2018**, por medio del cual se adopta el Estatuto de Propiedad Intelectual de la Universidad de Cundinamarca (Consejo Superior de la Universidad de Cundinamarca, 2018). Con base en este estatuto, se establecen las siguientes claridades:

- **Titularidad de los Estudiantes:** De acuerdo con el Artículo 16 del estatuto, la titularidad de los derechos de autor, tanto morales como patrimoniales, sobre el software desarrollado y el documento de tesis pertenece a los estudiantes autores, al ser una producción intelectual realizada en el marco de su trabajo de grado.
- **Reconocimiento de Coautoría:** En cumplimiento del mismo Artículo 16, se reconoce el derecho moral de la directora del proyecto a ser incluida como coautora de la obra, en virtud de su rol de orientación y asesoría en la investigación.

- **Licencia de Uso Académico para la Universidad:** Si bien la titularidad es de los estudiantes, se concede a la Universidad de Cundinamarca una licencia de uso no exclusiva y con fines académicos. Esto autoriza a la institución a incluir el trabajo de grado en sus bibliotecas y repositorios institucionales para fines de consulta, preservación e investigación, conforme a lo estipulado en los Artículos 14 y 24 del estatuto.

## Protección de Datos Personales

El sistema contempla la gestión de usuarios, por lo que su diseño y futura implementación se alinean con la **Ley Estatutaria 1581 de 2012**, por la cual se dictan disposiciones generales para la protección de datos personales (Congreso de la República de Colombia, 2012). El cumplimiento de esta ley se aborda desde dos perspectivas:

- **Fase Académica Actual:** En la etapa de desarrollo, el cumplimiento se centra en la adopción de buenas prácticas y en el diseño del sistema. Se garantiza el respeto a los principios de la ley, como la finalidad (los datos se usan solo para la autenticación), la seguridad de la información y el ejercicio de los derechos del titular a través de las funcionalidades que permiten al usuario gestionar sus propios datos.
- **Proyección a Futuro Uso Comercial:** Se reconoce que una eventual transición del sistema a un entorno comercial implicaría la asunción de obligaciones legales más estrictas, como el registro de la base de datos en el Registro Nacional de Bases de Datos (RNBD), la publicación de una política de tratamiento de datos y la implementación de canales formales para atender las solicitudes de los usuarios.

## **Seguridad Digital y Delitos Informáticos**

El desarrollo del software no solo se enfoca en la funcionalidad, sino también en la seguridad, considerando el marco de la **Ley 1273 de 2009**, que modifica el Código Penal en materia de delitos informáticos (Congreso de la República de Colombia, 2009). Las medidas de seguridad implementadas, como los controles de acceso y la autenticación de usuarios, tienen un doble propósito: proteger la integridad de los datos personales y prevenir de manera proactiva la comisión de conductas delictivas, como el acceso abusivo a un sistema informático. Este enfoque asegura que el proyecto se desarrolle de manera responsable y en conformidad con el marco legal que protege la información y los sistemas tecnológicos en el país.

## II. DOCUMENTACIÓN SOFTWARE

### 2.1. Plan de Proyecto

La gestión y planificación del proyecto se fundamentó en una adaptación del marco de trabajo ágil Scrum, tal como se describió en la sección de metodología. Este enfoque se eligió por su flexibilidad y su capacidad para adaptar el desarrollo a los hallazgos de la investigación de manera iterativa.

Para la gestión operativa de esta metodología, incluyendo la administración del *Product Backlog*, las historias de usuario y el seguimiento de los *Sprints*, se utilizó la herramienta de software de código abierto *Plane*.

La implementación de Scrum se adaptó a un contexto académico y de investigación, aplicando los siguientes elementos clave:

- **Historias de Usuario:** Todos los requerimientos funcionales del sistema, identificados en la primera fase del proyecto, se tradujeron en historias de usuario.
- **Product Backlog:** Se priorizaron las historias de usuario en un *Product Backlog*, que sirvió como la hoja de ruta principal para el desarrollo.
- **Sprints:** El trabajo de desarrollo se dividió en *Sprints*, con duraciones que oscilaron entre dos y tres semanas. Generalmente, cada *Product Backlog Item* (PBI) o conjunto de historias

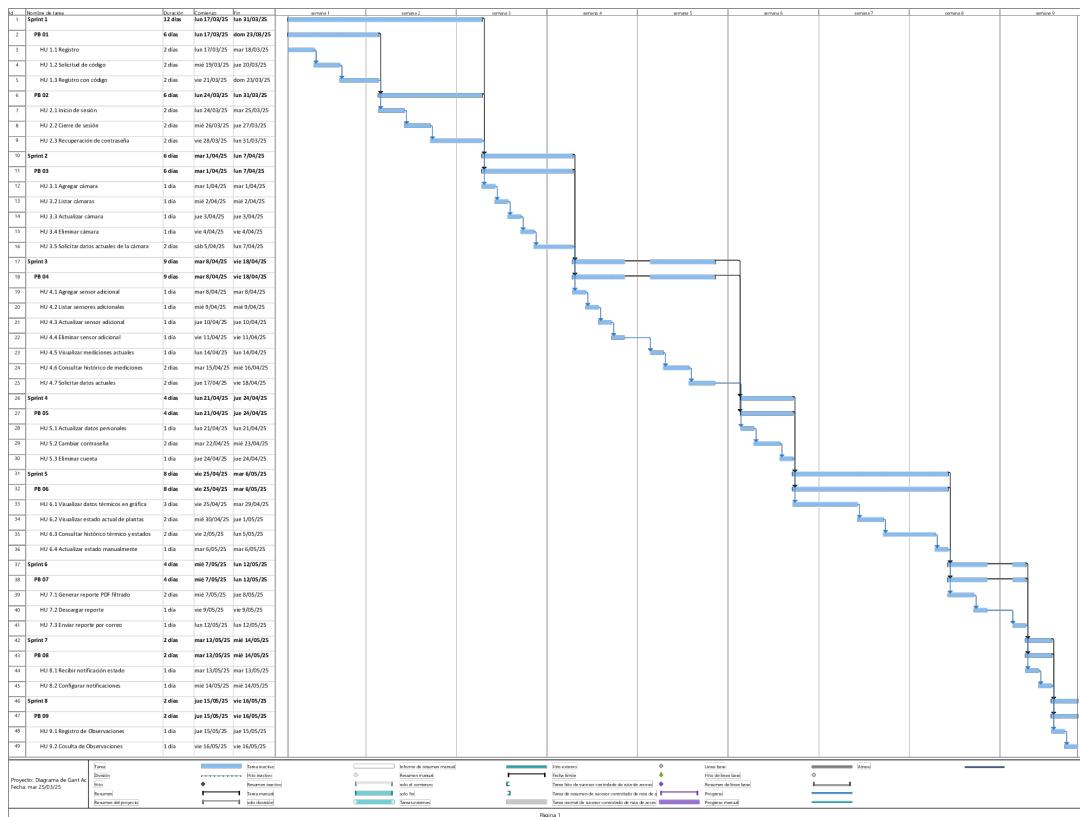
de usuario relacionadas se completaba dentro de un único *Sprint*.

- **Reuniones de Seguimiento:** Se realizaron reuniones semanales con la dirección del proyecto para revisar los adelantos, presentar el trabajo completado en el *Sprint* y ajustar las prioridades para el siguiente ciclo.

El cronograma detallado de ejecución del proyecto, que agrupa las tareas en los *Sprints* definidos, se presenta en el Diagrama de Gantt (Figura 1). Este diagrama ilustra la secuencia de las actividades, su duración y las dependencias entre las fases de investigación, desarrollo, integración y pruebas.

**Figura 1**

*Diagrama de Gantt del proyecto.*



## **2.2. Arquitectura del Software**

El presente capítulo detalla la arquitectura del software desarrollado para el sistema *Arandano IRT*. Se describe la estructura adoptada, los patrones de diseño implementados y las tecnologías seleccionadas, justificando las decisiones tomadas en función de los requerimientos funcionales y no funcionales del proyecto, así como de las lecciones aprendidas durante las fases iniciales de desarrollo.

El desarrollo del sistema partió de la construcción de un Prototipo Mínimo Viable (MVP), enfocado en validar las tecnologías centrales y las funcionalidades críticas de monitoreo y análisis. Esta fase inicial fue crucial, no solo para generar un conjunto de datos preliminar significativo (más de 3500 registros), sino también para identificar desafíos técnicos y refinar la arquitectura final. La experiencia con el MVP subrayó la importancia de una infraestructura resiliente. Un incidente específico, una interrupción prolongada del servicio debido a una falla en un proveedor externo (Cloudflare), motivó un rediseño orientado a minimizar las dependencias externas críticas y maximizar la autonomía operativa del sistema. Esta decisión estratégica ha resultado en una arquitectura robusta, evidenciada por un tiempo de actividad continuo superior a los 100 días desde su implementación final.

La arquitectura resultante se describe desde dos perspectivas complementarias: el diseño lógico, que aborda la organización interna del código y la aplicación del patrón Modelo-Vista-Controlador (MVC); y la arquitectura física, que detalla la distribución de los componentes de software en la infraestructura de despliegue. Finalmente, se presenta el stack (recursos) tecnológico que sustenta el sistema.

## 2.2.1. Patrón Arquitectónico y Diseño Lógico

El sistema *Arandano IRT* adopta el patrón arquitectónico Modelo-Vista-Controlador (MVC), aprovechando las capacidades ofrecidas por el framework **ASP.NET Core 8**. Se optó por una arquitectura monolítica modular en lugar de un enfoque desacoplado (como una Single Page Application (SPA) con una API backend separada), considerando el alcance del proyecto y la eficiencia en el desarrollo para un equipo reducido, decisión validada durante la fase del MVP. Esta elección permite una estructura cohesiva y un despliegue simplificado, sin sacrificar la organización interna del código.

La implementación sigue la interpretación clásica del patrón MVC adaptada al ecosistema de ASP.NET Core, estructurando la aplicación en las siguientes capas lógicas principales, tomadas de la Arquitectura Cebolla (Onion Architecture), reflejadas en la organización de directorios del proyecto:

- **Capa de Presentación (Vista - View):** Responsable de la interfaz de usuario (UI) y la interacción directa con el usuario (:Persona). Esta capa se implementa principalmente mediante **Razor Pages** y componentes **Blazor/Razor** (.cshtml, ubicados en el directorio /Views. Estos componentes se renderizan en el servidor, generando el HTML que se envía al navegador del cliente. La lógica de presentación, el manejo de eventos de UI y las llamadas iniciales a la lógica de negocio residen aquí.
- **Capa de Control (Controlador - Controller):** Actúa como intermediario entre la Vista y el Modelo. Los controladores (.cs), situados en 3\_Presentation/Controllers, reciben las solicitudes HTTP entrantes (generalmente iniciadas por acciones del usuario en la Vista), interpretan los datos de la solicitud, invocan la lógica de negocio necesaria a través de los servicios de la capa de aplicación y seleccionan la Vista apropiada para devolver la respuesta al cliente.

- **Capa de Modelo (Model):** Esta capa encapsula la lógica de negocio central, las reglas del dominio y el acceso a los datos. Se organiza en subcapas para una mejor separación de responsabilidades:
  - **Dominio (0\_Domain):** Contiene las entidades principales del sistema (ej. Crop, Plant, Device), objetos de valor, enumeraciones (Enums) y reglas de negocio fundamentales que son independientes de la tecnología.
  - **Aplicación (1\_Application):** Orquesta los casos de uso. Contiene los Data Transfer Objects (DTOs) utilizados para la comunicación entre capas, las interfaces de los servicios de aplicación (Contracts) y sus implementaciones (Implementation). Los servicios de esta capa coordinan la lógica de negocio, interactuando con las entidades del Dominio y utilizando la capa de Infraestructura para tareas como la persistencia o el envío de correos.
  - **Infraestructura (2\_Infrastructure):** Implementa los detalles técnicos y las dependencias externas. Incluye la configuración del **DbContext de Entity Framework Core** para la persistencia de datos en PostgreSQL, la implementación de servicios externos (como envío de correos con Brevo, almacenamiento con MinIO), la gestión de la autenticación, tareas en segundo plano (*background services*) y otros aspectos transversales. Proporciona las implementaciones concretas para las abstracciones definidas en las capas de Aplicación.

La comunicación entre estas capas se gestiona principalmente a través de Inyección de Dependencias (Dependency Injection), un principio fundamental en ASP.NET Core. Por ejemplo, un componente Razor en la Vista puede injectar y utilizar un servicio definido en la capa de Aplicación. Este servicio, a su vez, puede injectar y utilizar el **ApplicationDbContext** (definido en Infraestructura) para consultar o modificar entidades del Dominio persistidas en la base de datos. Este

flujo unidireccional de dependencias (Presentación → Aplicación → Dominio ← Infraestructura) asegura un bajo acoplamiento y facilita la mantenibilidad y testabilidad del sistema. La integración entre el frontend (Razor/Blazor) y el backend (Controladores, Servicios) se realiza mediante llamadas a métodos dentro del mismo proceso de aplicación, característico de la arquitectura monolítica seleccionada.

### 2.2.2. Arquitectura Física e Infraestructura

La arquitectura física define la distribución de los componentes de software en los nodos de hardware y la infraestructura subyacente que soporta la operación del sistema *Arandano IRT*. Esta arquitectura ha sido diseñada priorizando la robustez y la autonomía operativa, como se discutió en la introducción de este capítulo. La Figura 2 ilustra la topología de despliegue.

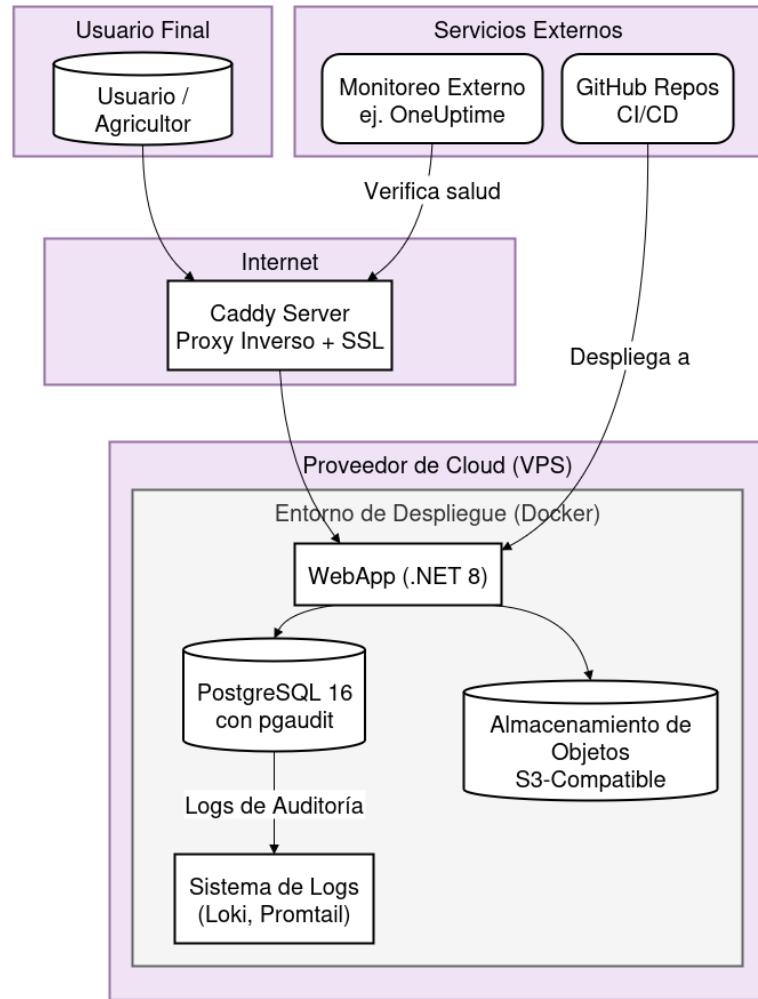
El sistema se despliega sobre dos nodos físicos principales:

1. **Servidor Central (Cloud VM):** Se utiliza una Máquina Virtual (VM) alojada en el proveedor de infraestructura como servicio (IaaS), específicamente DigitalOcean. Este nodo ejecuta el sistema operativo **Ubuntu Server 24.04 LTS** y alberga la totalidad de la pila de servicios backend, orquestada mediante Docker y Docker Compose. Los componentes de software que se ejecutan en contenedores dentro de esta VM incluyen:

- La aplicación principal **ASP.NET Core 8** (Backend MVC y API).
- El sistema gestor de base de datos **PostgreSQL 16**.
- El servicio de almacenamiento de objetos compatible con S3, **MinIO**, utilizado para las copias de seguridad de la base de datos y almacenamiento de archivos.
- La pila de observabilidad compuesta por **Loki/Promtail** para la agregación de logs y **Grafana** para la visualización y alertas.

**Figura 2**

*Diagrama de Despliegue Físico del Sistema Arandano IRT.*



- El proxy inverso **Caddy**, que gestiona el tráfico HTTPS entrante, la terminación TLS y el enrutamiento hacia los servicios correspondientes.

El servidor está protegido por firewalls a nivel de red (proveedor cloud) y a nivel de host (ufw), además de contar con Fail2Ban para la prevención de intrusiones.

2. **Dispositivo de Monitoreo (IoT Device):** Representa el hardware físico desplegado en campo cerca de las plantas de arándano. Este dispositivo integra los sensores ambientales y la

cámara térmica. Ejecuta un **firmware** embebido (desarrollado específicamente para el proyecto, como se detalla en el capítulo correspondiente) responsable de la captura periódica de datos (temperatura, humedad, imágenes térmicas y RGB), el preprocesamiento básico, y la transmisión segura de esta información al Servidor Central.

La comunicación entre estos nodos se realiza a través de protocolos de red estándar y seguros:

- **Dispositivo IoT → Servidor Central:** El firmware del dispositivo de monitoreo envía los datos recolectados (lecturas ambientales, estadísticas térmicas, logs) a la API expuesta por la aplicación ASP.NET Core 8 en el servidor. Esta comunicación se realiza exclusivamente sobre **HTTPS (puerto 443)**, asegurando el cifrado de la información en tránsito. El dispositivo se autentica ante la API mediante tokens de seguridad gestionados por el sistema .
- **Usuario (:Persona) → Servidor Central:** Los usuarios interactúan con la aplicación web a través de sus navegadores. Toda la comunicación entre el navegador del usuario y el servidor se realiza sobre **HTTPS (puerto 443)**, gestionada y asegurada por el proxy inverso Caddy, que maneja automáticamente los certificados TLS/SSL.

Esta arquitectura física centralizada en una VM gestionada con Docker simplifica el despliegue y mantenimiento, mientras que el uso de protocolos seguros garantiza la integridad y confidencialidad de los datos transmitidos, lo que ha permitido tener un sistema resiliente y confiable en operación continua.

### 2.2.3. Stack Tecnológico

La selección de las tecnologías para el sistema *Arandano IRT* se basó en criterios de soporte comunitario, costos y alineación con la filosofía *"open Source Primero"* del proyecto, sin descartar el uso pragmático de servicios privativos donde ofrecían ventajas significativas en sus capas gratuitas. A continuación, se detallan los componentes clave del stack tecnológico implementado:

- **Framework de Desarrollo Backend:** Se seleccionó **.NET 8** con el lenguaje **C#**. Esta elección se fundamenta en su naturaleza multiplataforma, su ecosistema moderno de desarrollo, el alto rendimiento ofrecido por la plataforma y el robusto soporte de la comunidad y Microsoft. El uso de Entity Framework Core como ORM facilitó la interacción con la base de datos.
- **Tecnologías de Frontend (UI):** La interfaz de usuario se construyó utilizando **Razor Pages** y componentes **Blazor/Razor** dentro del mismo proyecto ASP.NET Core. Esta aproximación simplifica el desarrollo y despliegue al mantener una base de código unificada y aprovechar las capacidades de renderizado del lado del servidor. Se complementa con HTML5, CSS3 y JavaScript para la interactividad del cliente.
- **Sistema Gestor de Base de Datos (SGBD):** Se optó por **PostgreSQL 16**. Esta decisión se basa en su reputación como un SGBD relacional de código abierto potente, fiable, extensible (uso de la extensión `pgaudit`) y con un excelente manejo de tipos de datos complejos como JSONB, utilizado en varias tablas del sistema.
- **Infraestructura de Despliegue y Orquestación:**
  - **Proveedor IaaS:** Se utiliza **DigitalOcean** como proveedor de la máquina virtual, debido a que ofrece un crédito gratuito a estudiantes mediante el Github Student Pack 2025.
  - **Sistema Operativo:** **Ubuntu Server 24.04 LTS** fue seleccionado como el SO base por su estabilidad, amplio soporte y compatibilidad con el ecosistema de contenedores.
  - **Contenerización:** **Docker** y **Docker Compose** son pilares fundamentales, permitiendo empaquetar, desplegar y gestionar los diferentes servicios de forma aislada y reproducible.
  - **Proxy Inverso:** **Caddy** se emplea para gestionar el tráfico entrante, el enruteamiento a los contenedores y, crucialmente, la automatización completa de la gestión de certificados

TLS/SSL.

- **Almacenamiento de Objetos:** **MinIO** proporciona una solución de almacenamiento compatible con S3, auto-hospedada, utilizada específicamente para las copias de seguridad de la base de datos y almacenamiento de archivos (imágenes).
- **Pila de Observabilidad:** Para el monitoreo y diagnóstico del sistema, se implementó una solución basada en:
  - **Agregación de Logs:** **Loki** centraliza los logs generados por todos los contenedores.
  - **Recolección de Logs:** **Promtail** se configura como el agente encargado de recolectar y enviar los logs desde los contenedores a Loki.
  - **Visualización y Alertas:** **Grafana** se utiliza para consultar los logs almacenados en Loki mediante LogQL, visualizar métricas y configurar alertas proactivas.
- **Servicios Externos Complementarios:** Aunque se priorizó el software auto-hospedado, se integraron servicios externos estratégicos como **Cloudflare** para la gestión avanzada de DNS y seguridad perimetral (WAF, Anti-DDoS, Anti-Bots), **Cloudflare Turnstile** como CAPTCHA no intrusivo, y **Brevo** para el envío fiable de notificaciones por correo electrónico transaccional certificado.

Esta combinación de tecnologías proporciona una base sólida, escalable y mantenible para el sistema, equilibrando el control ofrecido por las soluciones auto-hospedadas de código abierto con la conveniencia y especialización de algunos servicios externos privativos.

## 2.3. Determinación de Requerimientos

**Tabla 1**

*Requerimiento Funcional RF01: Registro*

<b>Identificador</b>	RF01
<b>Nombre</b>	Registro
<b>Roles</b>	Administrador, Usuario
<b>Descripción</b>	El administrador puede registrarse en el sistema de detección proporcionando los datos solicitados en el formulario de registro. Para que un usuario se pueda registrar, debe solicitar el código de acceso proporcionado por el administrador para poder realizar correctamente el registro.

**Tabla 2**

*Requerimiento Funcional RF02: Inicio de sesión*

<b>Identificador</b>	RF02
<b>Nombre</b>	Inicio de sesión
<b>Roles</b>	Administrador, Usuario
<b>Descripción</b>	Permite a los diferentes roles acceder al sistema de detección con sus credenciales (usuario y contraseña), estas deben ser correctas para su acceso. Al finalizar, se podrá cerrar sesión. En caso tal de olvidar la contraseña, se tendrá la opción para recuperarla.

**Tabla 3***Requerimiento Funcional RF03: CRUD cámara*

<b>Identificador</b> RF03	
<b>Nombre</b>	CRUD cámara
<b>Roles</b>	Administrador
<b>Descripción</b>	Se podrán agregar módulos termográficos al sistema de detección para poder recibir y procesar los datos que estas envíen. Además de visualizar y actualizar el estado de cada módulo termográfico. En caso de ser necesario, se podrá eliminar la cámara del sistema de detección.

**Tabla 4***Requerimiento Funcional RF04: CRUD persona*

<b>Identificador</b> RF04	
<b>Nombre</b>	CRUD persona
<b>Roles</b>	Administrador, Usuario
<b>Descripción</b>	Los distintos roles podrán actualizar sus datos personales o contraseña. También podrán eliminar su cuenta.

**Tabla 5***Requerimiento Funcional RF05: Módulo de mediciones*

<b>Identificador</b> RF05	
<b>Nombre</b>	Módulo de mediciones
<b>Roles</b>	Administrador, Usuario
<b>Descripción</b>	El sistema de detección recopilará y mostrará los datos de las mediciones tomadas por otros sensores por medio de gráficas y un histórico.

**Tabla 6***Requerimiento Funcional RF06: Módulo de procesamiento*

<b>Identificador</b> RF06	
<b>Nombre</b>	Módulo de procesamiento
<b>Roles</b>	Administrador, Usuario
<b>Descripción</b>	El sistema de detección mostrará los datos recopilados por los módulos de cámara por medio de gráficas. Además, se debe mostrar el estado de cada planta y el histórico de datos de todas las plantas. El procesamiento de los datos térmicos indicará el estado de cada planta. Los roles podrán actualizar el estado proporcionado por el sistema de ser necesario.

**Tabla 7***Requerimiento Funcional RF07: Reportes*

<b>Identificador</b> RF07	
<b>Nombre</b>	Reportes
<b>Roles</b>	Administrador, Usuario
<b>Descripción</b>	Los diferentes roles podrán generar reportes sobre el estado de las plantas en formato PDF con base en los datos recopilados por los módulos de cámara y/o por el módulo de procesamiento. Se podrá escoger distintos filtros (una o varias plantas, lapsos de tiempo).

**Tabla 8***Requerimiento Funcional RF08: Notificaciones*

<b>Identificador</b> RF08	
<b>Nombre</b>	Notificaciones
<b>Roles</b>	Administrador, Usuario (como receptores)
<b>Descripción</b>	Se deben enviar notificaciones por correo electrónico a los distintos roles en el cual se puedan alertar sobre cambios de estado en las plantas y enviar notificaciones de seguridad.

**Tabla 9***Requerimiento Funcional RF09: Gestionar Observaciones*


---

<b>Identificador</b>	RF09
<b>Nombre</b>	Gestionar Observaciones
<b>Roles</b>	Administrador, Usuario
<b>Descripción</b>	<p>Permite a los usuarios registrar y consultar observaciones cualitativas sobre el estado de las plantas.</p> <p>Se utiliza una plantilla estandarizada para anotar aspectos visuales (color, textura, uniformidad, daños), asignar una calificación subjetiva y añadir notas, complementando los datos cuantitativos para la metodología mixta y el procesamiento de datos.</p>

---

**Tabla 10***Requerimiento No Funcional RNF01: Seguridad*


---

<b>Identificador</b>	RNF01
<b>Nombre</b>	Seguridad
<b>Roles</b>	N/A (Aplica al Sistema)
<b>Descripción</b>	El sistema de detección debe cumplir con los lineamientos y leyes establecidos para la protección, integridad y disponibilidad de los datos (Ley 1581 de 2012).

---

**Tabla 11**

*Requerimiento No Funcional RNF02: Copia de seguridad*

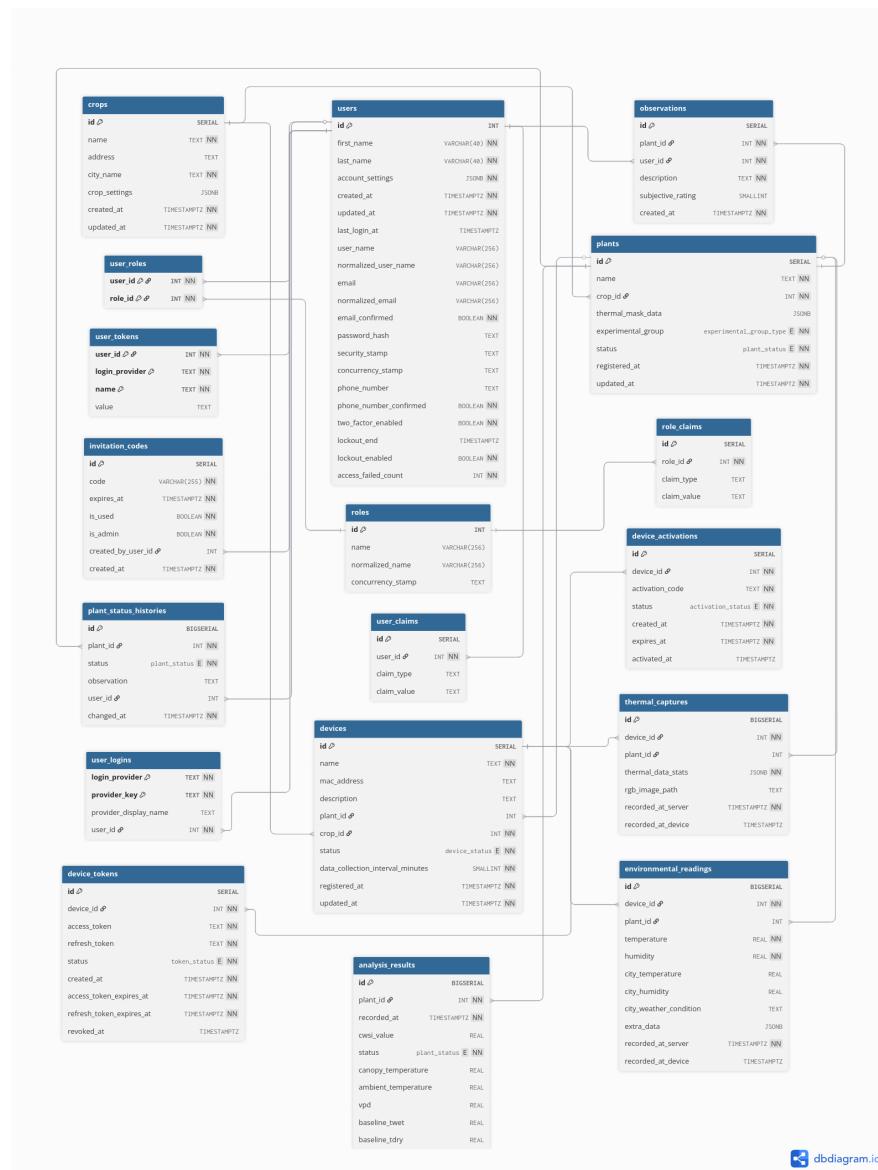
<b>Identificador</b> RNF02	
<b>Nombre</b>	Copia de seguridad
<b>Roles</b>	N/A (Aplica al Sistema)
<b>Descripción</b>	Se debe asegurar un respaldo de los datos en caso de presentarse alguna eventualidad no se vulnere la integridad de los datos. Esta copia de seguridad se debe hacer de forma automática y semanal.

## 2.4. Especificación del Diseño

### 2.4.1. Modelo de Entidad-Relación (MER)

**Figura 3**

*Diagrama Entidad-Relación del Sistema.*



La estructura de la base de datos diseñada para el sistema *Arandano IRT* organiza la información necesaria para el funcionamiento de la aplicación web y la gestión de los datos recolectados. Las tablas se agrupan funcionalmente para facilitar su comprensión, como se describe a continuación.

### **Grupo 1: Tipos Enumerados (ENUMs)**

Este grupo define tipos de datos personalizados que garantizan la consistencia y restringen los valores posibles para campos clave relacionados con estados o clasificaciones dentro del sistema. Estos tipos son fundamentales para mantener la integridad referencial semántica en diversas tablas.

- `device_status`: Cataloga los estados operativos de los dispositivos físicos de monitoreo.
- `activation_status`: Define las fases del proceso de activación de un nuevo dispositivo.
- `token_status`: Indica la validez de los tokens de autenticación para dispositivos.
- `plant_status`: Especifica los posibles estados de estrés hídrico detectados o asignados a una planta.
- `experimental_group_type`: Clasifica las plantas según su rol en configuraciones experimentales (ej. Control, Estrés, Monitoreado).

### **Grupo 2: Tablas Núcleo**

Constituyen las entidades centrales del dominio del sistema, representando los elementos fundamentales del monitoreo.

- `crops`: Representa cada unidad de cultivo o lote. Funciona como la entidad agrupadora principal, almacenando información geográfica y parámetros de configuración específicos del cultivo.

- **users**: Almacena la información de los usuarios registrados en la aplicación web, incluyendo datos personales, credenciales de acceso gestionadas por ASP.NET Core Identity y configuraciones de cuenta personalizadas.
- **plants**: Modela cada planta individual bajo monitoreo. Se vincula a un cultivo (**crop\_id**) y registra su estado de estrés hídrico actual (**status**), su clasificación experimental (**experimental\_group**) y datos asociados a la máscara térmica utilizada para el análisis de estrés hídrico.

### **Grupo 3: Identidad y Autenticación de Usuarios (ASP.NET Core Identity)**

Este conjunto de tablas implementa el esquema estándar de ASP.NET Core Identity, gestionando de forma robusta la autenticación, autorización basada en roles y la seguridad de las cuentas de usuario de la aplicación web.

- **roles, user\_roles**: Definen los roles disponibles en el sistema y establecen la relación muchos-a-muchos entre usuarios y roles.
- **user\_claims, role\_claims**: Permiten la asignación de permisos granulares (claims) directamente a usuarios o a roles, facilitando un modelo de autorización flexible.
- **user\_logins, user\_tokens**: Gestionan la integración con proveedores de identidad externos y el almacenamiento de tokens para funciones como la confirmación de correo o el restablecimiento de contraseña.
- **invitation\_codes**: Administra un sistema de registro basado en invitaciones, controlando la generación, validez y uso de códigos únicos para la creación de nuevas cuentas de usuario, vinculando opcionalmente la invitación a quien la generó.

## **Grupo 4: Gestión y Autenticación de Dispositivos**

Este grupo se enfoca en el registro, seguimiento del estado y la autenticación segura de los dispositivos de hardware (sensores, cámaras térmicas) responsables de la recolección de datos en campo.

- **devices:** Registra cada dispositivo físico, asociándolo a un cultivo y opcionalmente a una planta específica. Almacena metadatos como el nombre, descripción, estado operativo y la frecuencia configurada para la recolección de datos.
- **device\_activations:** Facilita el proceso seguro de incorporación (*onboarding*) de nuevos dispositivos al sistema mediante códigos de activación de un solo uso, registrando el estado y temporalidad de dicho proceso.
- **device\_tokens:** Almacena los tokens que permiten a los dispositivos autenticarse ante la API del sistema para el envío seguro de datos, gestionando su ciclo de vida (activo/revocado) y fechas de expiración.

## **Grupo 5: Recolección y Análisis de Datos**

Este grupo es fundamental, ya que almacena las mediciones directas de los sensores, las observaciones cualitativas y los resultados derivados del procesamiento y análisis realizado por el sistema.

- **environmental\_readings:** Persiste las series temporales de datos ambientales (temperatura, humedad) recolectadas por cada dispositivo, incluyendo timestamps tanto del dispositivo como del servidor y opcionalmente, datos climáticos externos contextuales.
- **thermal\_captures:** Almacena los metadatos y estadísticas clave extraídas de cada captura termográfica (temperaturas mínima, máxima, promedio, histograma), así como la referencia

a la imagen RGB asociada (almacenada externamente). Incluye timestamps del dispositivo y del servidor.

- **observations:** Permite a los usuarios registrar evaluaciones cualitativas manuales sobre el estado de las plantas, proporcionando un complemento a los datos cuantitativos de los sensores. Se vincula a la planta observada y al usuario que realizó el registro.
- **plant\_status\_histories:** Funciona como un log de auditoría específico para el estado de las plantas. Registra cada cambio en el **plant\_status**, indicando la fecha, el nuevo estado, la planta afectada y si el cambio fue realizado manualmente por un usuario o automáticamente por el sistema tras un análisis.
- **analysis\_results:** Guarda los resultados cuantitativos generados por los algoritmos de análisis del sistema, como el valor calculado del Índice de Estrés Hídrico (CWSI). Almacena el resultado junto con la fecha, la planta analizada y otros parámetros relevantes para la trazabilidad del cálculo, asegurando la unicidad por planta y timestamp.

#### **2.4.2. Diagramas de Casos de Uso**

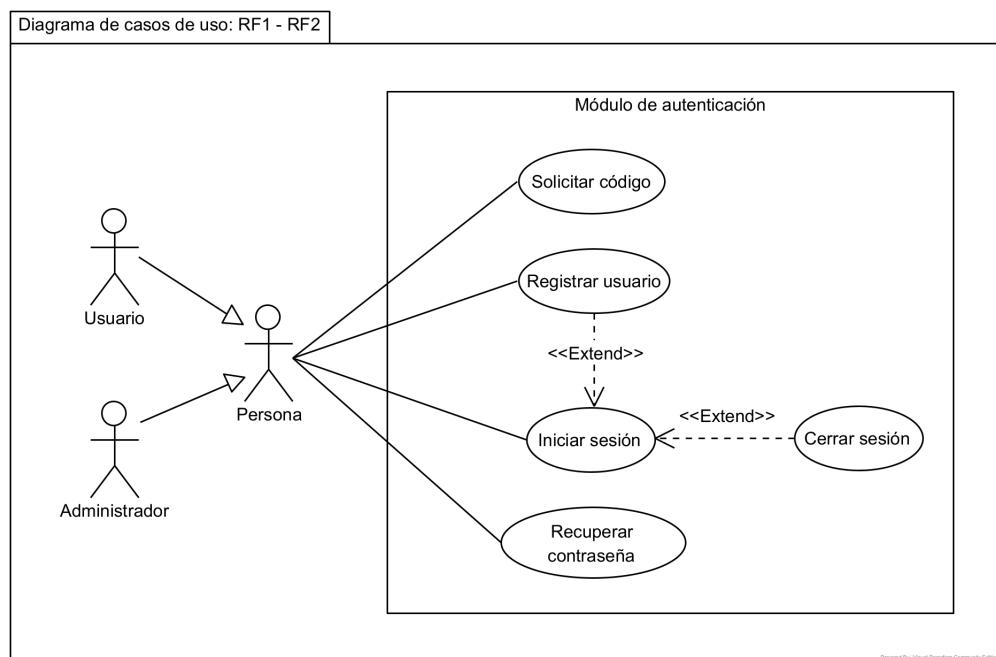
Un caso de uso es una unidad coherente de funcionalidad externamente visible proporcionada por un clasificador (denominado sistema) y expresada mediante secuencias de mensajes intercambiados por el sistema y uno o más actores de la unidad del sistema (Rumbaugh, Jacobson, y Booch, 2007). El propósito de un caso de uso es definir una pieza de comportamiento coherente sin revelar la estructura interna del sistema (Rumbaugh y cols., 2007).

Esta sección presenta los diagramas de casos de uso que modelan las funcionalidades principales ofrecidas por el sistema. Estos diagramas ilustran, desde una perspectiva de alto nivel, cómo interactúan los diferentes actores principalmente el **Usuario**, el **Administrador** y, en ciertos escenarios, el propio **Sistema** con las funcionalidades clave del sistema (representadas por elipses).

Cada diagrama está generalmente asociado a un módulo funcional o a un Requerimiento Funcional (RF) específico identificado en la especificación de requisitos, utilizando la notación estándar UML para representar actores, casos de uso, límites del sistema y relaciones como «Extend» o «Include». El objetivo es proporcionar una visión clara del alcance funcional del sistema desde el punto de vista de sus usuarios.

**Figura 4**

*Diagrama de Casos de Uso para la Gestión de Usuarios (RF1, RF2).*



La Figura 4 detalla los casos de uso correspondientes al Módulo de autenticación del sistema, cubriendo las funcionalidades de registro e inicio de sesión (RF01 y RF02). Los actores que interactúan con este módulo son el Usuario y el Administrador, ambos representados mediante la generalización Persona. A continuación se describe cada caso de uso:

### **Caso de Uso: Solicitar código**

Permite a un **Usuario** (actuando como **Persona**) solicitar un código de acceso. Según RF01, este código es necesario para que un **Usuario** pueda registrarse y asociarse a un cultivo existente, y debe ser proporcionado previamente por un **Administrador**.

### **Caso de Uso: Registrar usuario**

Corresponde a la funcionalidad RF01. Permite a **Persona** crear una nueva cuenta en el sistema. El flujo varía según el rol: un **Administrador** puede registrarse directamente, mientras que un **Usuario** necesita ingresar un código de acceso válido (obtenido a través del caso de uso **Solicitar código**) para completar su registro dentro de un cultivo específico.

### **Caso de Uso: Iniciar sesión**

Representa la funcionalidad principal de RF02, permitiendo a **Persona** acceder al sistema mediante la validación de sus credenciales (correo electrónico y contraseña). Un inicio de sesión exitoso otorga acceso a las funcionalidades correspondientes al rol del usuario (**Administrador** o **Usuario**). Este caso de uso es la base para poder interactuar con el resto del sistema y puede ser extendido por **Cerrar sesión**.

### **Caso de Uso: Recuperar contraseña**

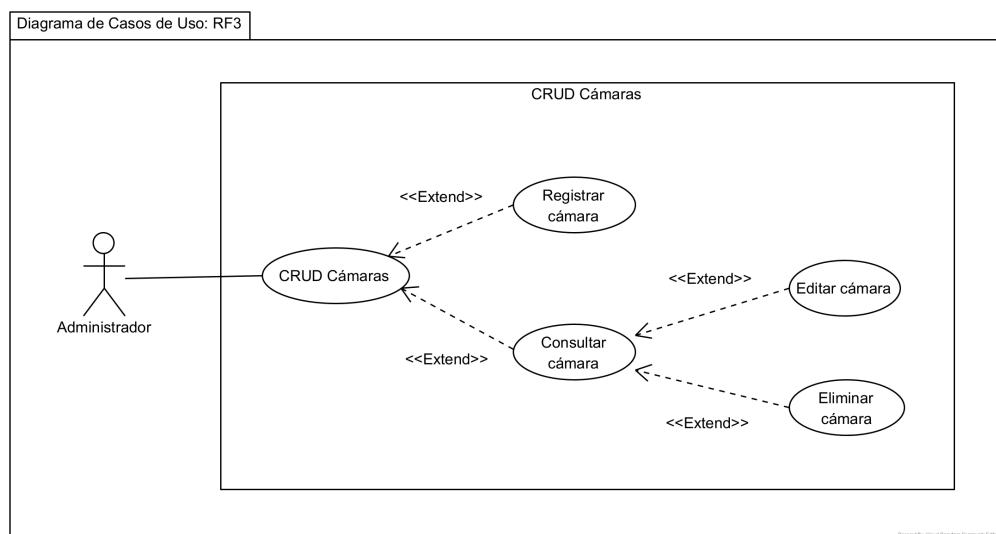
Forma parte de la funcionalidad RF02. Ofrece a **Persona** un mecanismo para restablecer su contraseña si la ha olvidado. Típicamente, esto implica un proceso de verificación a través del correo electrónico registrado para garantizar la seguridad.

## Caso de Uso: Cerrar sesión

Este caso de uso extiende (‘«Extend»’) a **Iniciar sesión**, para completar su funcionamiento. Representa la acción explícita y opcional que realiza **Persona** para terminar de forma segura su sesión activa dentro de la aplicación, después de haber iniciado sesión y realizado otras tareas.

**Figura 5**

*Diagrama de Casos de Uso para la Gestión de Cámaras (RF3).*



La Figura 5 describe los casos de uso asociados a la gestión (CRUD - Crear, Leer, Actualizar, Borrar) de los dispositivos de cámara o módulos termográficos, funcionalidad identificada como RF03 y exclusiva para el actor **Administrador**. El diagrama presenta un caso de uso central y las operaciones específicas que extienden su funcionalidad:

## Caso de Uso: CRUD Cámaras

Representa la funcionalidad principal o el punto de acceso para que el **Administrador** gestione los módulos termográficos registrados en el sistema. Este caso de uso, descrito en RF03, se ve extendida (‘«Extend»’) por operaciones más específicas como registrar o consultar cámaras.

### **Caso de Uso: Registrar cámara**

Extiende ('«Extend»') la funcionalidad de CRUD Cámaras. Permite al Administrador añadir un nuevo módulo termográfico al sistema (operación Create). Esto incluye la configuración inicial del dispositivo dentro de la plataforma.

### **Caso de Uso: Consultar cámara**

Extiende ('«Extend»') la funcionalidad de CRUD Cámaras. Permite al Administrador buscar y visualizar la información detallada y el estado actual de los módulos termográficos ya registrados en el sistema (operación Read). Este caso de uso sirve como punto de partida para otras acciones opcionales.

### **Caso de Uso: Editar cámara**

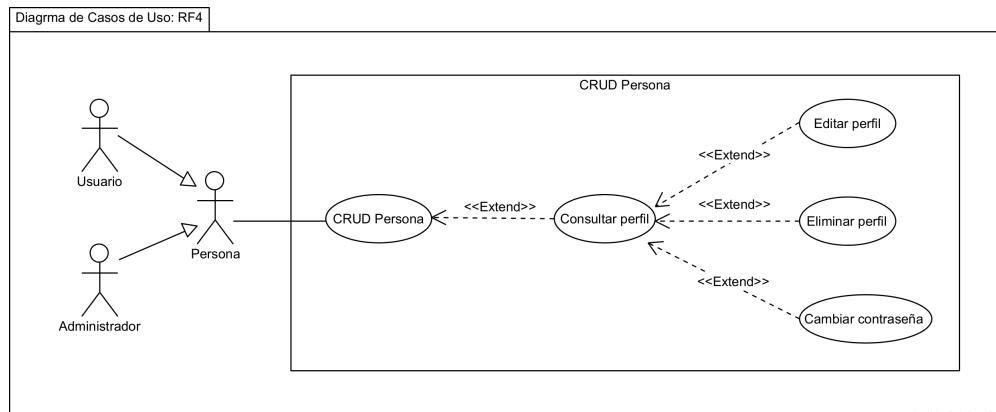
Extiende ('«Extend»') la funcionalidad de Consultar cámara. Una vez que el Administrador ha consultado los detalles de una cámara específica, tiene la opción de modificar su configuración, nombre, descripción o actualizar su estado dentro del sistema (operación Update).

### **Caso de Uso: Eliminar cámara**

Extiende ('«Extend»') la funcionalidad de Consultar cámara. Después de consultar o seleccionar una cámara, el Administrador puede optar por eliminar permanentemente el registro de ese dispositivo del sistema (operación Delete), usualmente si el dispositivo se da de baja o ya no se utiliza.

**Figura 6**

*Diagrama de Casos de Uso para la Gestión de Perfiles (RF4).*



La Figura 6 detalla los casos de uso relacionados con la gestión del perfil de usuario dentro del sistema, funcionalidad descrita en RF04. Estas operaciones pueden ser realizadas tanto por el **Usuario** como por el **Administrador**, representados por la generalización **Persona**, sobre la información de su propia cuenta. El diagrama se centra en el módulo o funcionalidad **CRUD Persona**:

### Caso de Uso: CRUD Persona

Este caso de uso actúa como el punto de entrada general para que **Persona** administre la información asociada a su perfil en el sistema, tal como se indica en RF04. La funcionalidad principal que extiende (‘«Extend»’) esta gestión es **Consultar perfil**.

### Caso de Uso: Consultar perfil

Extiende (‘«Extend»’) la funcionalidad de **CRUD Persona**. Permite a **Persona** visualizar sus propios datos de perfil registrados en la aplicación (operación de Leer). Esta consulta es, generalmente, el paso previo necesario para poder realizar modificaciones o eliminar la cuenta.

### **Caso de Uso: Editar perfil**

Extiende (‘«Extend»’) la funcionalidad de Consultar perfil. Una vez que Persona visualiza su perfil, este caso de uso le permite modificar sus datos personales registrados, como nombre, apellido, correo electrónico o preferencias de notificación (operación de Actualizar datos), de acuerdo con RF04.

### **Caso de Uso: Eliminar perfil**

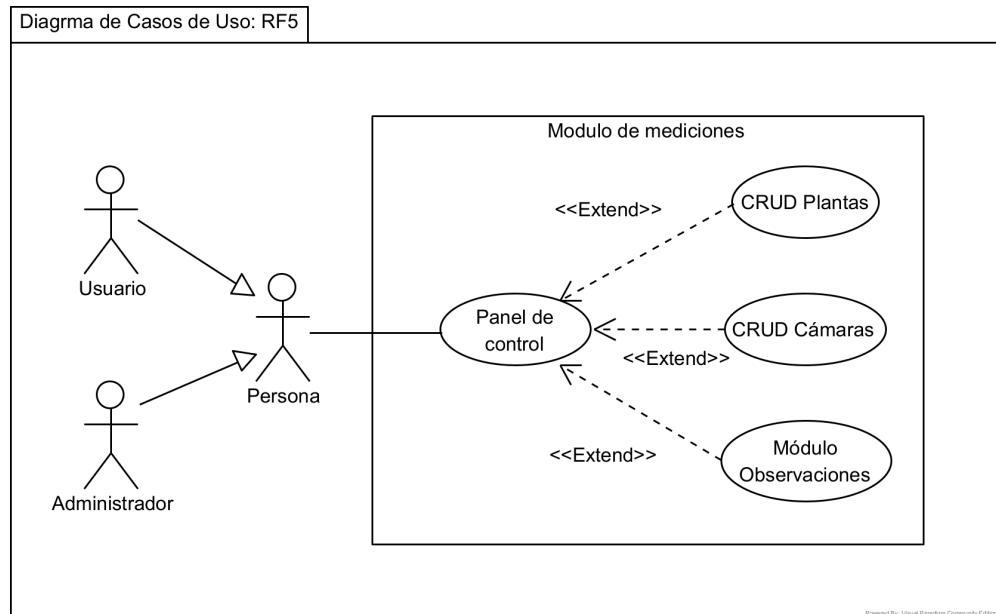
Extiende (‘«Extend»’) la funcionalidad de Consultar perfil. Habilita a Persona para solicitar la eliminación permanente de su cuenta y datos asociados del sistema (operación de Borrar), como lo permite RF04. Esta acción se realiza típicamente desde la vista del perfil del usuario.

### **Caso de Uso: Cambiar contraseña**

Extiende (‘«Extend»’) la funcionalidad de Consultar perfil. Permite a Persona iniciar el proceso para actualizar su contraseña de acceso al sistema (operación de Actualizar contraseña), como se menciona en RF04. Usualmente, esta opción está disponible dentro de la sección de gestión o consulta del perfil.

**Figura 7**

Diagrama de Casos de Uso para el Módulo de Mediciones (RF5).



La Figura 7 presenta los casos de uso asociados al **Modulo de mediciones** del sistema. Este módulo, accesible por los actores **Usuario** y **Administrador** (generalizados como **Persona**), funciona como el panel de control principal o *Dashboard* tras el inicio de sesión. A continuación, se describen los casos de uso involucrados:

### Caso de Uso: Panel de control

Este caso de uso representa la pantalla principal que visualiza **Persona** al interactuar con el módulo de mediciones. Su función primordial, relacionada con RF05, es mostrar de forma consolidada los datos clave recopilados por los sensores y cámaras, típicamente mediante gráficas de las últimas 24 horas y los últimos valores registrados. Proporciona una visión general del estado del cultivo y sirve como punto central desde el cual se puede acceder ('«Extend»') a otras funcionalidades específicas de gestión.

### **Caso de Uso: CRUD Plantas**

Extiende ('«Extend»') la funcionalidad del Panel de control. Permite a Persona gestionar las plantas registradas en el sistema: añadir nuevas plantas, consultar su información, editar sus detalles o eliminarlas (Crear, Leer, Actualizar, Borrar). Esta es una funcionalidad esencial para administrar las entidades principales monitoreadas (PlantData).

### **Caso de Uso: CRUD Cámaras**

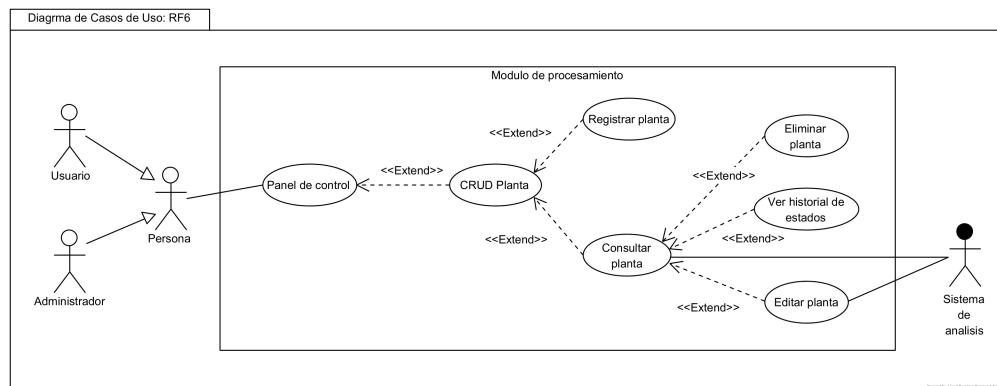
Extiende ('«Extend»') la funcionalidad del Panel de control, proporcionando un acceso a la gestión de los dispositivos (módulos termográficos y sensores). Es importante resaltar que, aunque el Panel de control es accesible por Persona, la funcionalidad específica de CRUD Cámaras está restringida al rol Administrador, tal como se definió en RF03. Permite al Administrador realizar las operaciones de Crear, Leer, Actualizar y Borrar sobre los dispositivos desde este panel.

### **Caso de Uso: Módulo Observaciones**

Extiende ('«Extend»') la funcionalidad del Panel de control. Representa la funcionalidad añadida para gestionar las observaciones cualitativas realizadas sobre las plantas. Permite a Persona registrar y consultar notas descriptivas sobre aspectos visuales (decoloraciones, uniformidad, notas sobre hojas/tallos) o calificaciones subjetivas del estado de la planta. Esta funcionalidad es clave para cumplir con la metodología de investigación y permitir una mejor compresión de los datos recopilados.

**Figura 8**

*Diagrama de Casos de Uso para la Gestión de Plantas (RF6).*



La Figura 8 ilustra los casos de uso pertenecientes al **Modulo de procesamiento**, cuya funcionalidad principal (descrita en RF06) es la gestión de las plantas y la visualización de su estado y datos procesados. Los actores involucrados son el **Usuario** y el **Administrador** (generalizados como **Persona**), además de un actor externo, el **Sistema de análisis**.

### Caso de Uso: Panel de control

Reutilizado del diagrama anterior (Figura 7), sirve como punto de entrada general para **Persona**, desde donde se puede acceder ('«Extend»') a la funcionalidad específica de gestión de plantas (**CRUD Planta**).

### Caso de Uso: CRUD Planta

Actúa como el caso de uso central para la administración del ciclo de vida de las plantas dentro de este módulo. Es accedido desde el **Panel de control** y engloba las operaciones fundamentales sobre las plantas, siendo extendido ('«Extend»') por acciones más específicas como **Registrar planta** y **Consultar planta**.

### **Caso de Uso: Registrar planta**

Extiende ('«Extend»') la funcionalidad de CRUD Planta. Permite a Persona añadir una nueva planta al sistema (operación Create), registrando su información inicial para comenzar el monitoreo.

### **Caso de Uso: Consultar planta**

Extiende ('«Extend»') la funcionalidad de CRUD Planta. Permite a Persona visualizar (Leer) la información detallada de una planta específica. Principalmente, según RF06, mostrar su estado actual (resultado del procesamiento de datos). Este caso de uso también es utilizado por el Sistema de análisis externo, para consultar el estado o datos procesados de las plantas. Sirve como base para otras operaciones extendidas.

### **Caso de Uso: Editar planta**

Extiende ('«Extend»') la funcionalidad de Consultar planta. Permite a Persona modificar (Actualizar) la información asociada a una planta existente. Como se especifica en RF06, esto incluye la capacidad de los usuarios para actualizar manualmente el estado asignado a la planta si lo consideran necesario tras una revisión. Este caso de uso también es utilizado por el Sistema de análisis externo, para actualizar el estado de una planta después de procesar los datos obtenidos de los sensores y cámaras.

### **Caso de Uso: Eliminar planta**

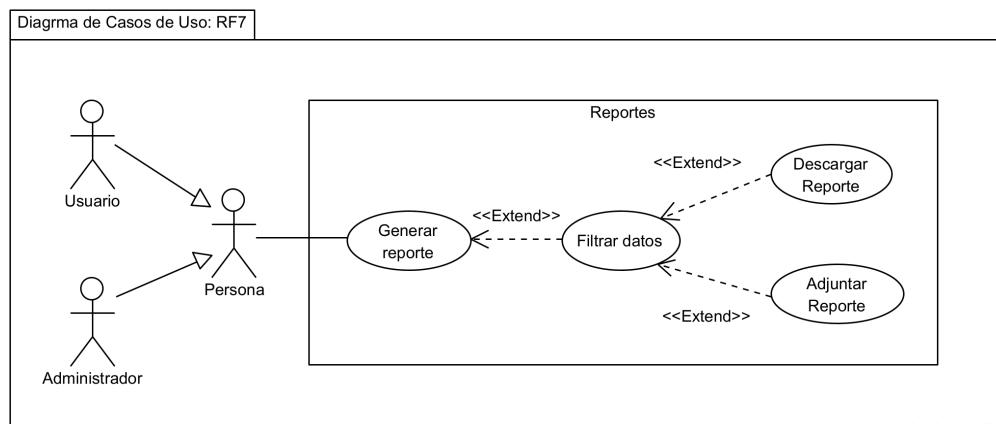
Extiende ('«Extend»') la funcionalidad de Consultar planta. Otorga a Persona la capacidad de eliminar (Borrar) el registro de una planta del sistema, por ejemplo, si la planta física es retirada del cultivo.

## Caso de Uso: Ver historial de estados

Extiende ('«Extend»') la funcionalidad de Consultar planta. Permite a Persona acceder y visualizar el registro histórico de los cambios de estado que ha tenido una planta a lo largo del tiempo, funcionalidad explícitamente mencionada en RF06 para el seguimiento de la condición de las plantas.

**Figura 9**

*Diagrama de Casos de Uso para la Generación de Reportes (RF7).*



La Figura 9 presenta los casos de uso correspondientes a la generación de Reportes del sistema, funcionalidad descrita en RF07. Tanto el Usuario como el Administrador (generalizados como Persona) pueden acceder a estas funcionalidades para obtener informes sobre el estado de las plantas y otros módulos del sistema.

## Caso de Uso: Generar reporte

Este es el caso de uso principal que inicia Persona para solicitar la creación de un informe. Principalmente, el sistema compila la información relevante sobre el estado de las plantas basándose en los datos recopilados y procesados. El resultado es un reporte consolidado que se genera en formato PDF.

### **Caso de Uso: Filtrar datos**

Extiende ('«Extend»') la funcionalidad de Generar reporte. Proporciona a Persona la opción de aplicar criterios específicos para refinar el contenido del informe antes de su generación final. Por ejemplo, filtrar por planta(s) específica(s) o por lapsos de tiempo.

### **Caso de Uso: Descargar Reporte**

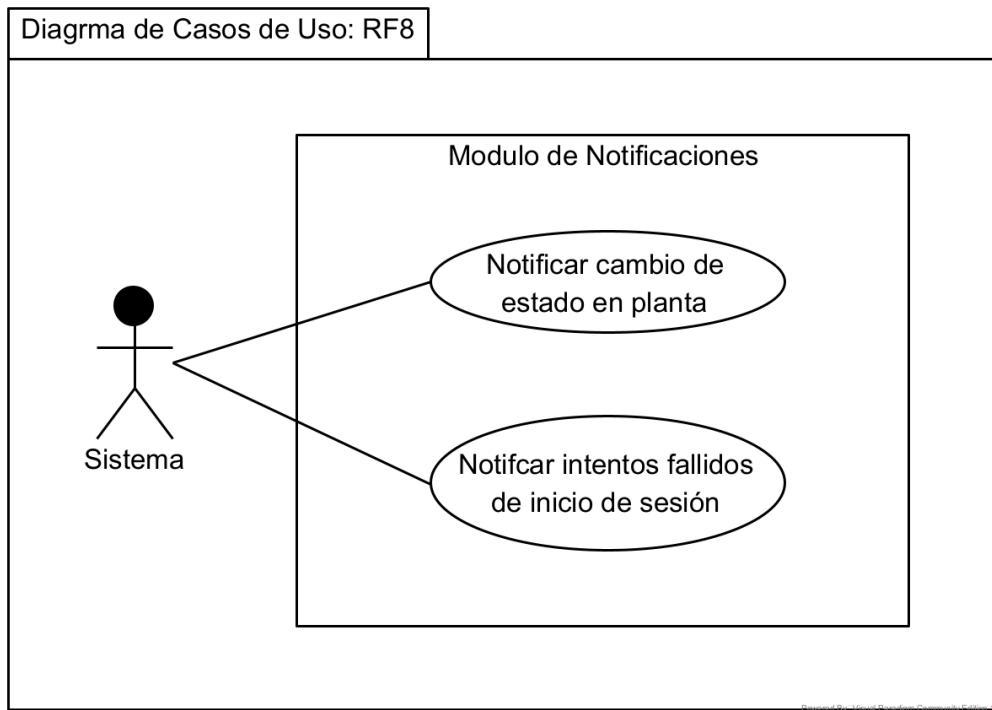
Extiende ('«Extend»') la funcionalidad de Filtrar datos (y, por lo tanto, la de Generar reporte). Una vez que el reporte ha sido generado (y posiblemente filtrado), este caso de uso permite a Persona descargar el archivo resultante (en formato PDF) a su dispositivo local para su consulta o almacenamiento.

### **Caso de Uso: Adjuntar Reporte**

Extiende ('«Extend»') la funcionalidad de Filtrar datos. Representa una opción disponible después de generar (y filtrar) el reporte. Permite de adjuntar el reporte generado por medio de un correo electrónico del usuario autenticado.

**Figura 10**

*Diagrama de Casos de Uso para las Notificaciones (RF8).*



La Figura 10 ilustra los casos de uso del **Modulo de Notificaciones**, que corresponden a la funcionalidad descrita en RF08. En este módulo, el actor principal que inicia las acciones es el propio **Sistema**, indicando que estas notificaciones son procesos automatizados. Estas alertas se envían por correo electrónico a los roles **Administrador** y **Usuario** del sistema.

### **Caso de Uso: Notificar cambio de estado en planta**

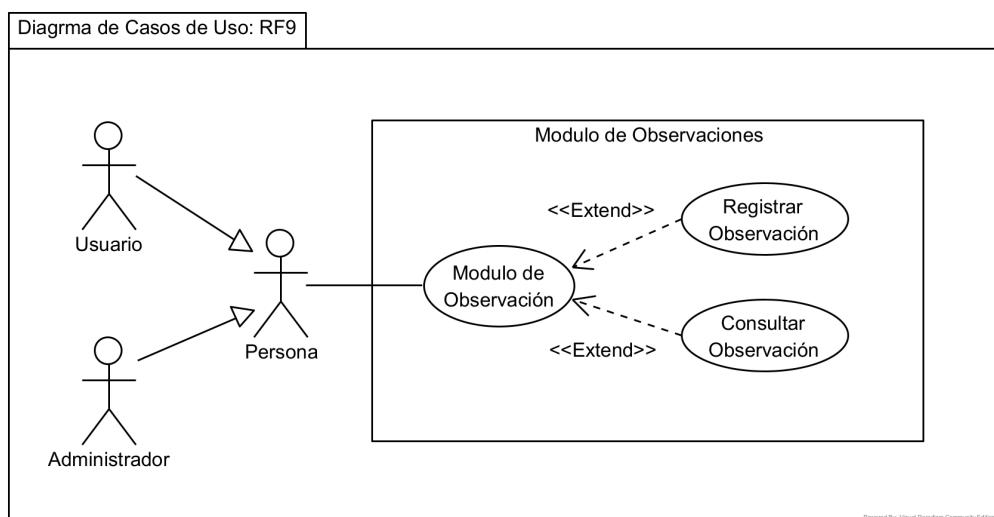
Este caso de uso es ejecutado automáticamente por el **Sistema**. Se activa cuando se produce un cambio significativo en el estado registrado de una planta (por ejemplo, al pasar de 'Saludable' a 'No Saludable' o viceversa, basado en el procesamiento de datos o una actualización manual). El objetivo es alertar proactivamente a los usuarios relevantes (**Administrador**, **Usuario**) por correo electrónico sobre la nueva condición de la planta, facilitando una respuesta rápida.

## Caso de Uso: Notificar intentos fallidos de inicio de sesión

Iniciado también por el Sistema, este caso de uso forma parte de las "notificaciones de seguridad" mencionadas en RF08. Se activa cuando el sistema detecta una actividad potencialmente sospechosa, como múltiples intentos fallidos de inicio de sesión asociados a una cuenta de usuario. El propósito es informar al usuario afectado, mediante correo electrónico, sobre estos intentos para que pueda verificar la seguridad de su cuenta y tomar acciones si es necesario (ej. cambiar contraseña).

**Figura 11**

*Diagrama de Casos de Uso para el Módulo de Observaciones (RF9).*



La Figura 11 describe los casos de uso para el **Modulo de Observaciones**. Esta funcionalidad, descrita en RF09, es esencial para la metodología mixta del proyecto, permitiendo la recolección de datos cualitativos sobre las plantas. Los actores **Usuario** y **Administrador** (generalizados como **Persona**) interactúan con este módulo.

### **Caso de Uso: Modulo de Observación**

Este caso de uso representa el punto de entrada principal para que Persona interactúe con las funcionalidades de registro y consulta de observaciones cualitativas de las plantas. Sirve como interfaz para acceder a las operaciones específicas que extienden ('«Extend»') su funcionalidad.

### **Caso de Uso: Registrar Observación**

Extiende ('«Extend»') la funcionalidad del Modulo de Observación. Permite a Persona registrar una nueva observación cualitativa sobre una planta específica, utilizando la plantilla definida en el diseño experimental. Esto incluye seleccionar el estado general visual, describir cambios, anotar aspectos específicos de color/textura y hojas/tallos, asignar una calificación subjetiva y opcionalmente notas adicionales. Estos datos son cruciales para complementar los datos cuantitativos y tener una mejor comprensión de los datos.

### **Caso de Uso: Consultar Observación**

Extiende ('«Extend»') la funcionalidad del Modulo de Observación. Permite a Persona buscar y visualizar las observaciones cualitativas previamente registradas para una planta. Esto facilita el seguimiento de la evolución visual descrita en el diseño experimental y la comparación con los datos cuantitativos (termografía, sensores).

#### **2.4.3. Diagramas de Secuencia**

Un diagrama de secuencia muestra un conjunto de mensajes ordenados en una secuencia temporal (Rumbaugh y cols., 2007). Cada rol se muestra como una línea de vida es decir, una línea vertical que representa al rol a lo largo del tiempo a través de la interacción completa (Rumbaugh y cols., 2007). Los mensajes se muestran con flechas entre líneas de vida (Rumbaugh y cols., 2007).

## Líneas de Vida Principales

Los diagramas de secuencia presentados en este documento ilustran las interacciones entre diferentes componentes del sistema para realizar casos de uso específicos. A continuación, se presenta una descripción general de las líneas de vida (*lifelines*) que aparecen de forma recurrente, representando los actores y las capas arquitectónicas principales del sistema, el cual sigue el patrón Modelo-Vista-Controlador (MVC) y organizado mediante carpetas que simulan las capas de la arquitectura cebolla (Onion Architecture).

- **:Persona:** Representa al usuario final que interactúa con el sistema a través de la interfaz gráfica. Dependiendo del contexto, puede ser un **Usuario** o un **Administrador**. Es el iniciador de las secuencias asociadas a funcionalidades interactivas.
- **:Vista:** Simboliza la interfaz de usuario (capa *View* en MVC) con la que interactúa **:Persona**. Esta línea de vida representa la aplicación cliente (ej. aplicación web ejecutándose en el navegador). Sus responsabilidades incluyen presentar información al usuario, capturar sus entradas, realizar validaciones del lado del cliente y enviar solicitudes (vía HTTP/S) a los puntos de entrada del backend (**:Controlador**), así como renderizar las respuestas recibidas.
- **:Controlador:** Representa la capa de Controladores (*Controller* en MVC) en el backend. Actúa como el punto de entrada para las solicitudes HTTP provenientes de **:Vista**. Es responsable de interpretar la solicitud, invocar la lógica de negocio apropiada interactuando con el **:Modelo** (a través de servicios), coordinar las operaciones necesarias y seleccionar la **:Vista** adecuada (o devolver datos, como API) para generar la respuesta al cliente.
- **:Modelo:** Simboliza la capa del Modelo (*Model* en MVC). Encapsula los datos de la aplicación (entidades de dominio como Planta, Cultivo, Usuario), la lógica de negocio fundamental, las reglas de validación y la interacción con la capa de persistencia de datos. El **:Controlador**

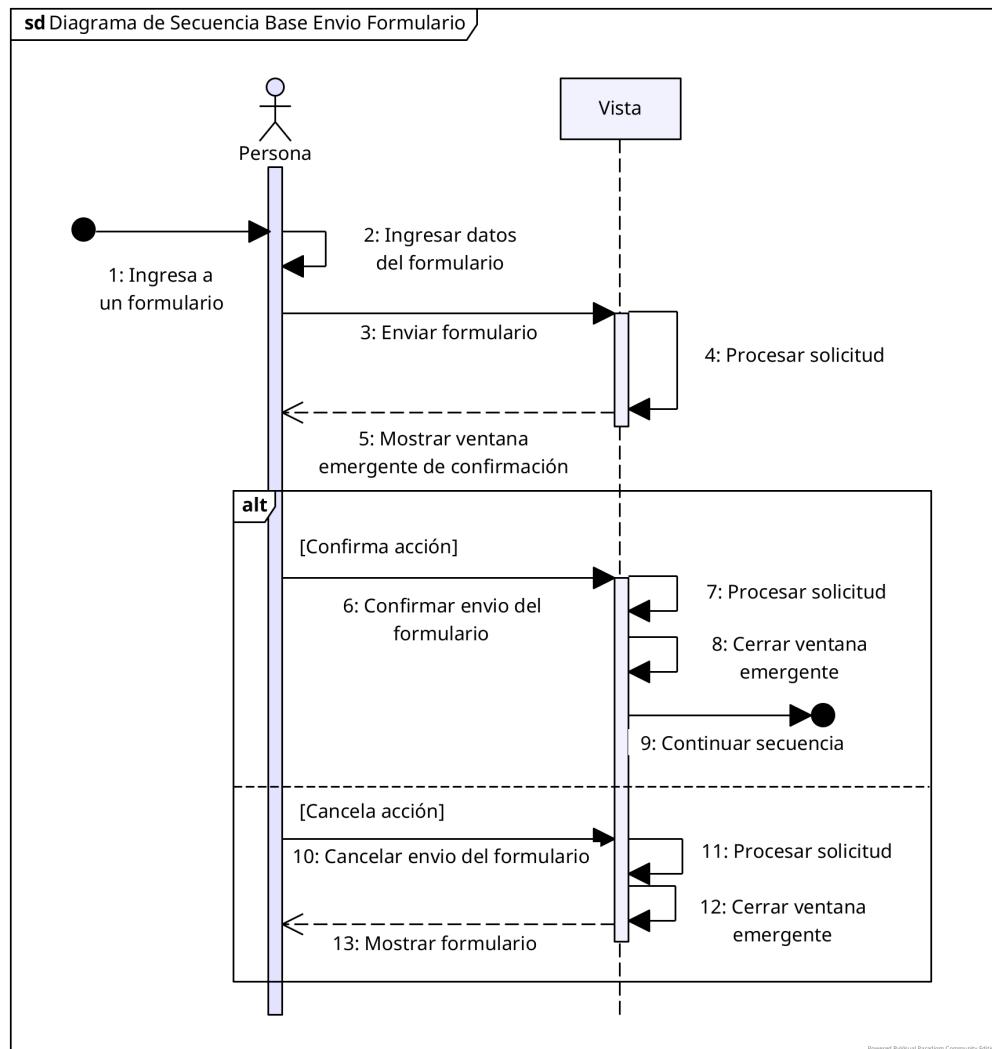
utiliza el :Modelo para leer o modificar el estado de la aplicación. La interacción directa con la base de datos (PostgreSQL) se realiza a través del ORM Entity Framework Core, orquestado mediante servicios, utilizando las entidades definidas en el Modelo.

Generalmente, el flujo de una solicitud iniciada por el usuario sigue la secuencia :Persona → :Vista → :Controlador. El :Controlador interactúa entonces con el :Modelo (capas de servicio que usan el Modelo) para procesar la solicitud y acceder a los datos. Finalmente, el :Controlador selecciona una :Vista (o prepara una respuesta de datos) que se envía de vuelta a :Persona a través de la :Vista original: :Controlador → :Vista → :Persona. Esta estructura promueve la separación de responsabilidades característica del patrón MVC.

Es importante señalar que aquellos diagramas que introducen líneas de vida con roles particulares no cubiertos en la descripción general (como la línea de vida :Camara detallada en la Figura19) o aquellos que representan patrones de interacción fundamentales y reutilizables (como el flujo base para envío de formularios mostrado en la Figura12) incluyen una descripción textual específica adjunta. Para los diagramas de secuencia restantes, se entiende que siguen el patrón general de interacción entre :Vista, :Controlador y :Modelo, utilizando las responsabilidades asignadas a cada línea de vida según se describió previamente.

**Figura 12**

Diagrama de secuencia base: Envío de formulario.



### Líneas de Vida Involucradas

Las líneas de vida principales en este diagrama base son:

- : **Persona**: Representa al usuario (Usuario o Administrador) que interactúa con la interfaz gráfica del sistema. Es quien inicia la acción, ingresa los datos y toma la decisión final de confirmar o cancelar el envío.

- :Vista: Representa el componente de la interfaz de usuario (la pantalla o ventana específica) que contiene el formulario. Recibe los datos ingresados, gestiona el proceso de envío inicial y maneja el diálogo de confirmación con el usuario.

## Descripción del Flujo Genérico

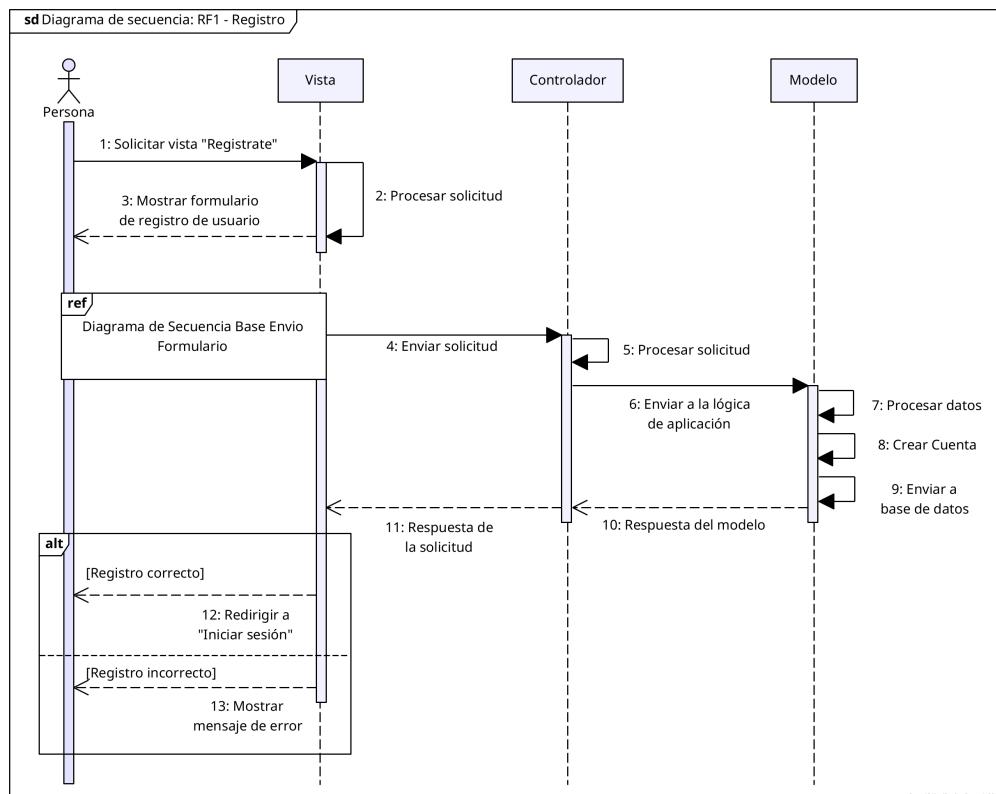
La secuencia describe el siguiente flujo general:

1. El proceso comienza cuando :Persona accede a un formulario (1: **Ingresar a un formulario**) e introduce los datos requeridos (2: **Ingresar datos del formulario**).
2. :Persona inicia la acción de envío (3: **Enviar formulario**) hacia la :Vista.
3. La :Vista realiza un procesamiento inicial (4: **Procesar solicitud**), que podría incluir validaciones del lado del cliente.
4. La :Vista solicita una confirmación explícita al usuario mostrando una ventana emergente (5: **Mostrar ventana emergente de confirmación**).
5. Se presenta un fragmento alternativo (**alt**) basado en la respuesta de :Persona:
  - **Si [Confirma acción]**: :Persona confirma el envío (6). La :Vista procede con el procesamiento definitivo de la solicitud (7), cierra la ventana emergente (8) y permite continuar la secuencia (9), lo que usualmente implica una redirección o un mensaje de éxito. (*Nota: El paso 7 es donde, en diagramas específicos, se detallaría la comunicación con controladores, servicios y base de datos*).
  - **Si [Cancela acción]**: :Persona cancela el envío (10). La :Vista procesa la cancelación (11), cierra la ventana emergente (12) y vuelve a mostrar el formulario (13), permitiendo al usuario corregir datos o abandonar la tarea.

Este diagrama establece la interacción fundamental usuario-interfaz para operaciones de formulario, haciendo énfasis en el paso de confirmación antes del procesamiento final.

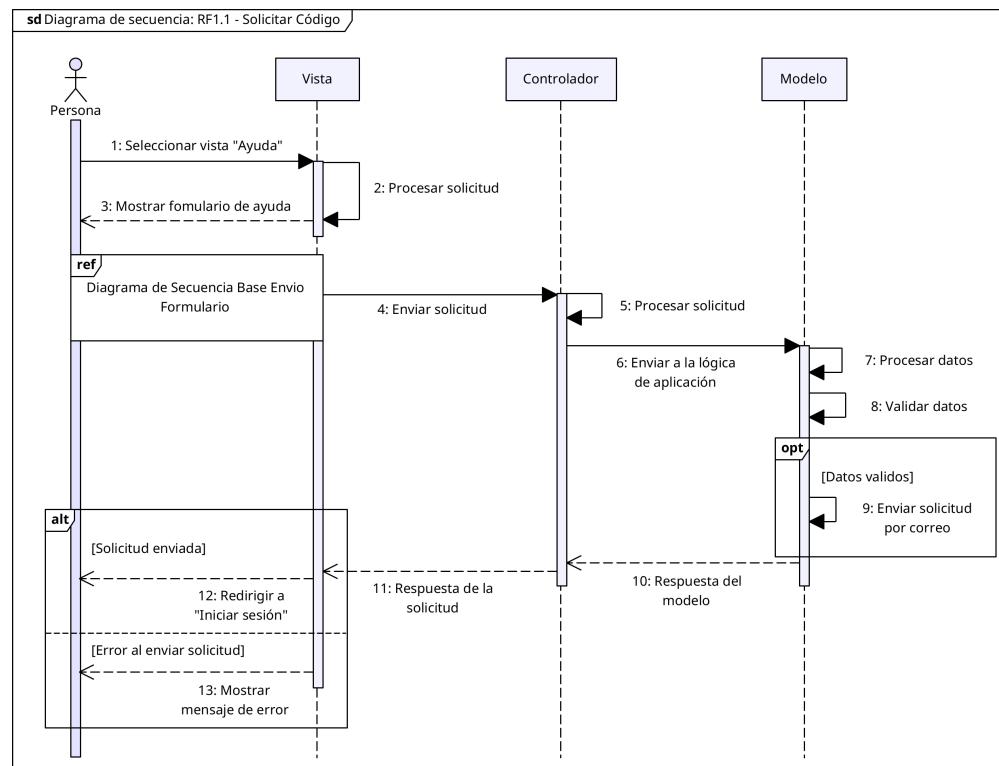
**Figura 13**

*Diagrama de Secuencia para el Registro (RF1.0).*



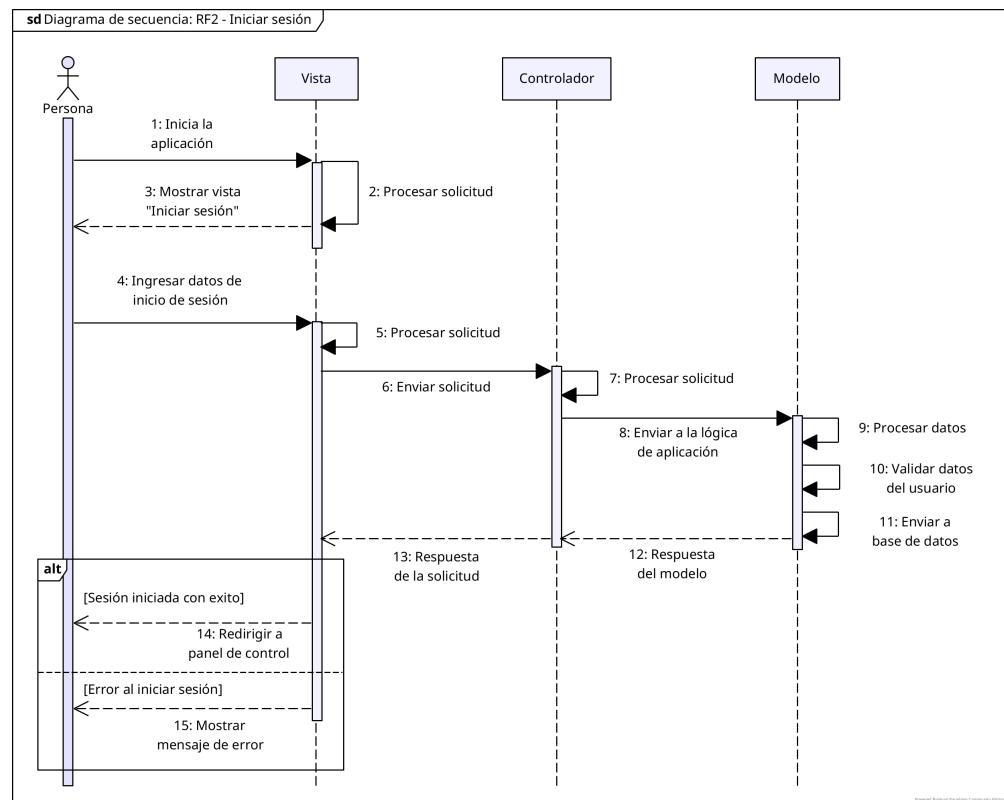
**Figura 14**

Diagrama de Secuencia para Solicitar Código (RF1.1).



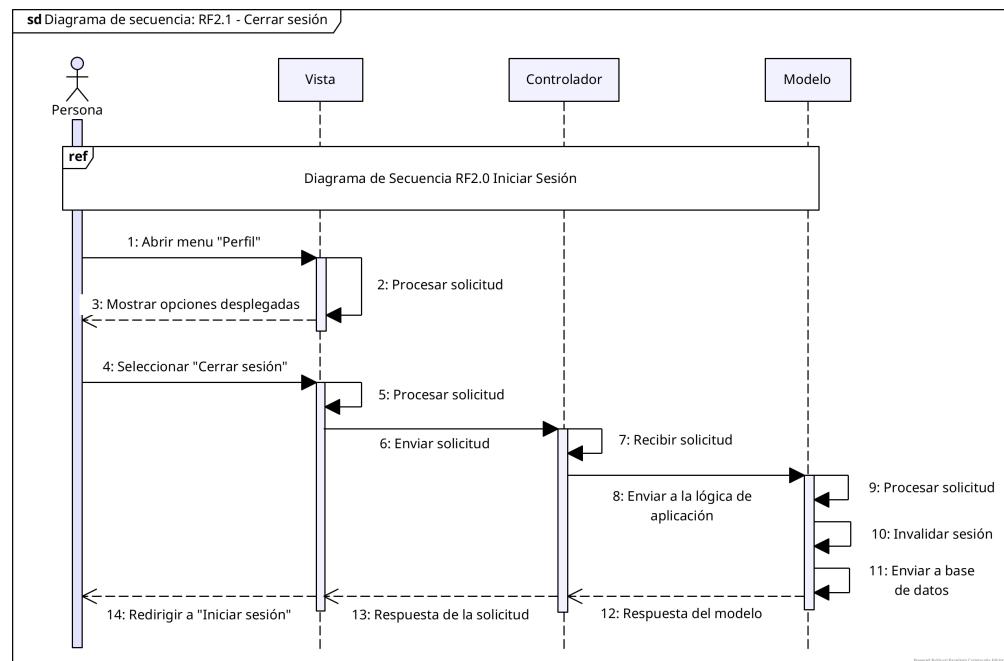
**Figura 15**

Diagrama de Secuencia para Iniciar Sesión (RF2.0).



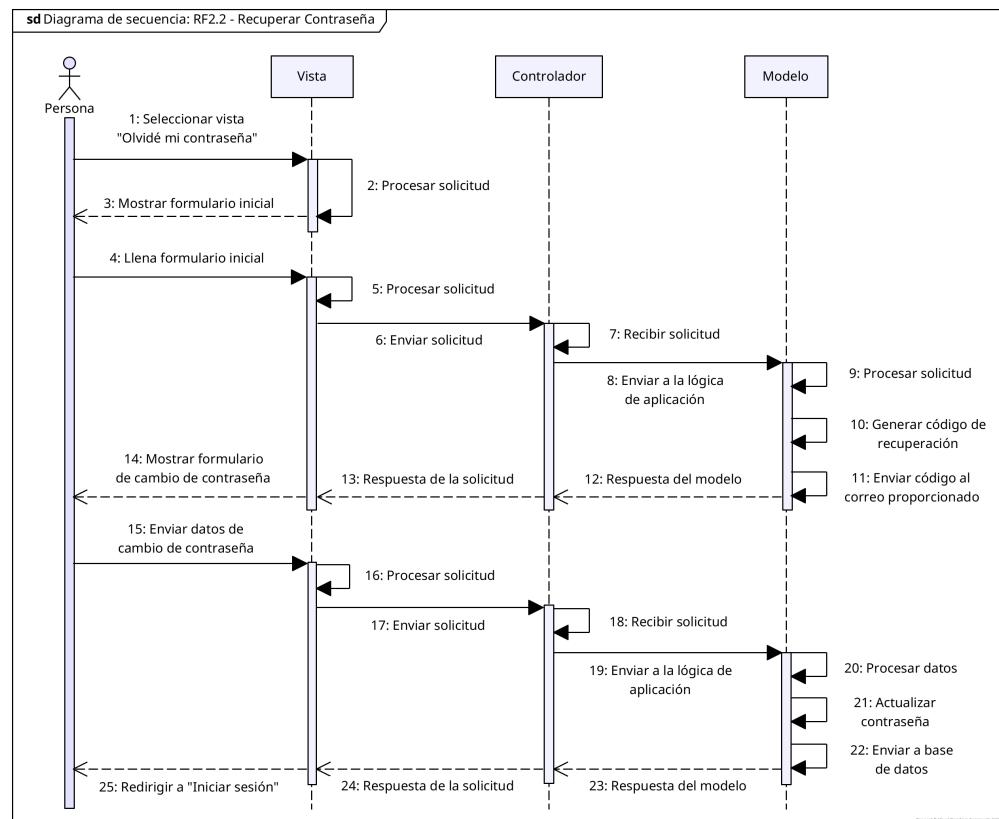
**Figura 16**

Diagrama de Secuencia para Cerrar Sesión (RF2.1).



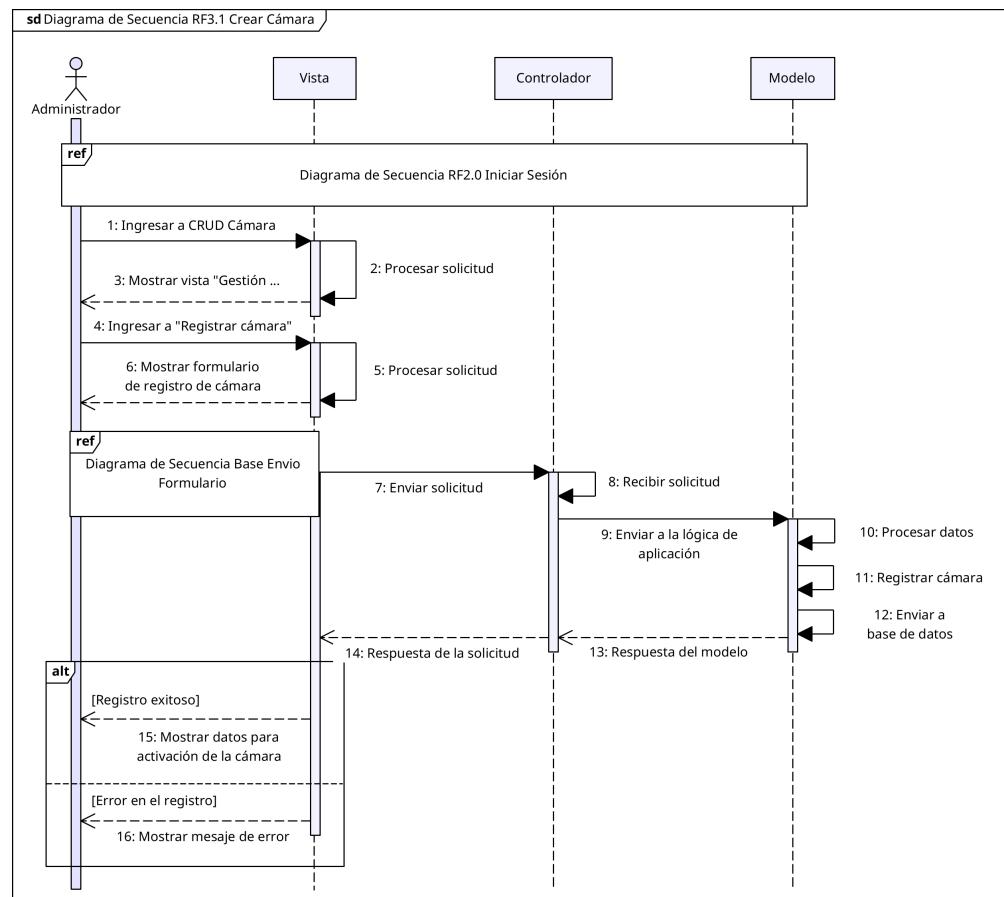
**Figura 17**

Diagrama de Secuencia para Recuperar Contraseña (RF2.2).



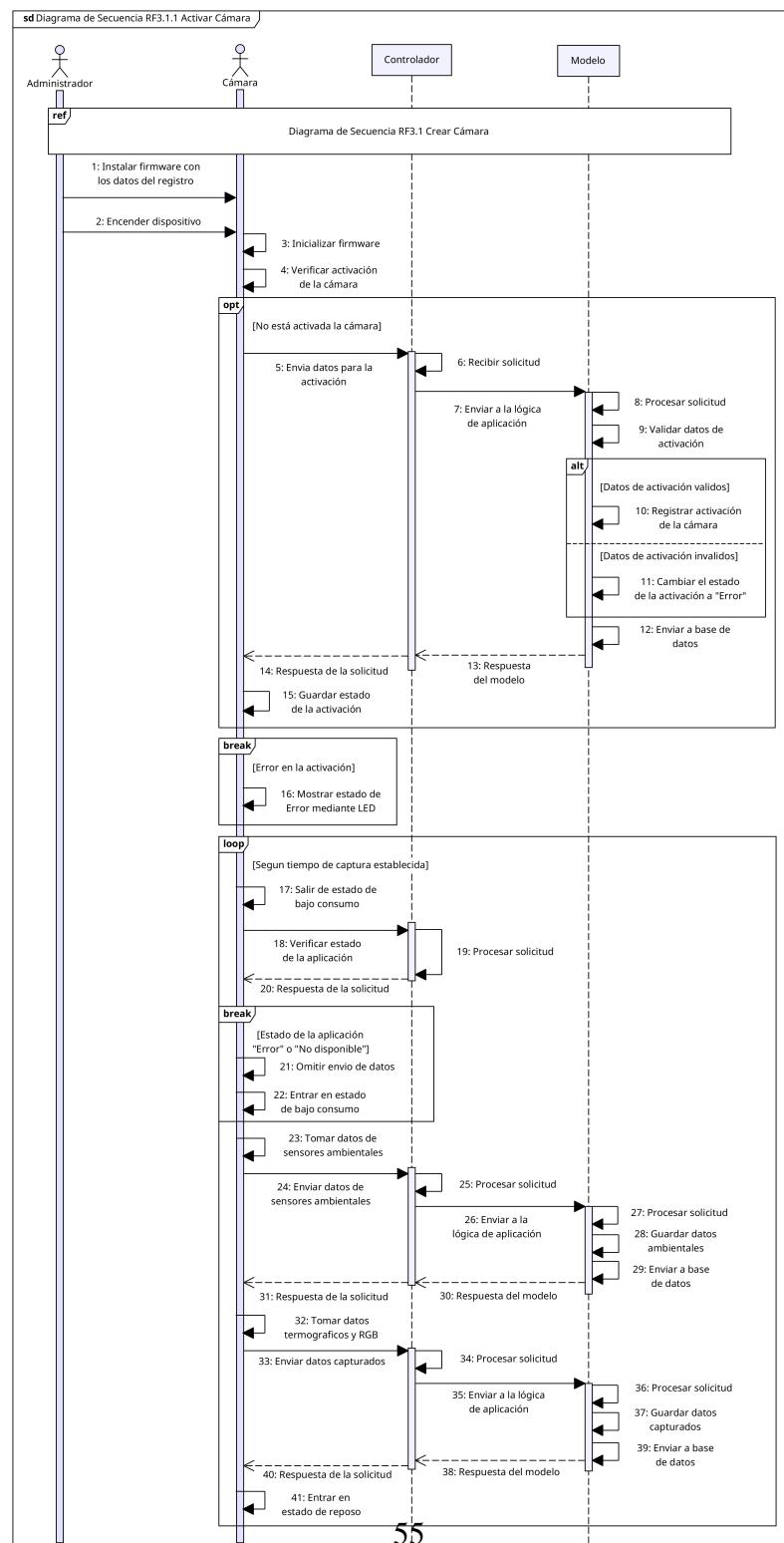
**Figura 18**

Diagrama de Secuencia para Crear Cámara (RF3.1).



**Figura 19**

Diagrama de Secuencia para Activar Cámara (RF3.1.1).



## Línea de Vida :Camara

La responsabilidad principal de la línea de vida :Camara es interactuar con la API del sistema. Encapsula la lógica autónoma del dispositivo físico, manejando su activación, configuración, el ciclo periódico de toma y envío de datos (ambientales y de imagen), y la gestión básica de errores de comunicación con la API. Su comportamiento se divide en dos fases principales:

### 1. Fase de Activación:

- Al iniciar y verificar la conexión de red, la :Camara envía un código de activación único a la API.
- Espera una respuesta de la API. Si la activación es exitosa, recibe la configuración operativa (ej. intervalo de muestreo) que fue definida por el Administrador durante el registro del dispositivo en el sistema.
- Almacena esta configuración recibida de forma persistente (memoria no volátil) para guiar su funcionamiento futuro.
- Si el proceso de activación falla (ej. código inválido, API no responde correctamente), la :Camara entra en un estado de error y detiene el proceso, sin pasar a la fase operativa.

### 2. Fase Operativa (Ciclo de Trabajo):

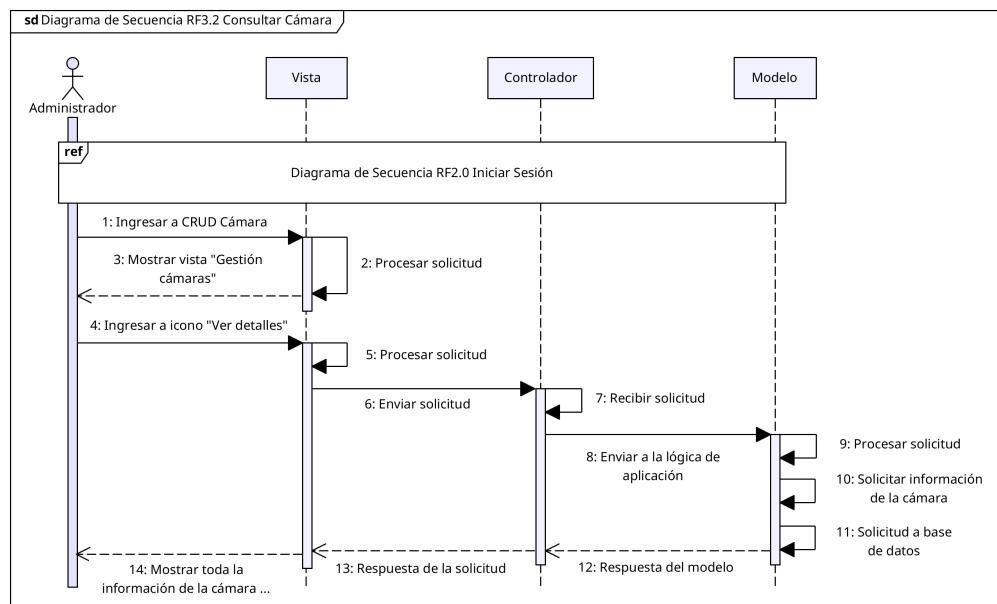
- Una vez activada, la :Camara opera en un ciclo continuo basado en el intervalo de tiempo definido en su configuración almacenada.
- Al inicio de cada ciclo, verifica la disponibilidad de la API.
- **Si la API está disponible:** Procede a recolectar los datos de los sensores ambientales (temperatura, humedad, etc.) y los envía a la API. Seguidamente, captura las imágenes (termográfica

y RGB) y las envía también a la API. Después de enviar los datos, entra en un modo de bajo consumo o reposo hasta que el temporizador del ciclo indique el inicio del siguiente.

- **Si la API no está disponible:** La :Camara omite la recolección y envío de datos para ese ciclo. Entra en un periodo de espera antes de volver a intentar la verificación de la API en el siguiente ciclo programado.

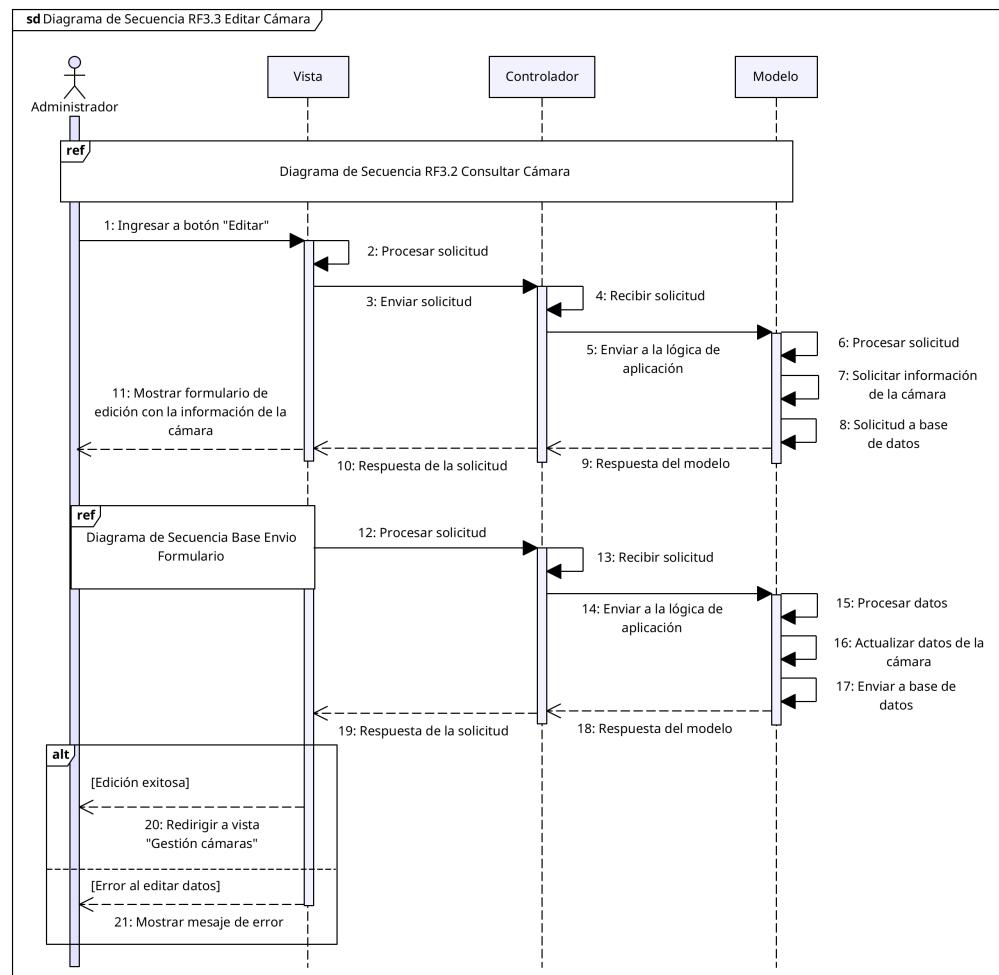
**Figura 20**

*Diagrama de Secuencia para Consultar Cámara (RF3.2).*



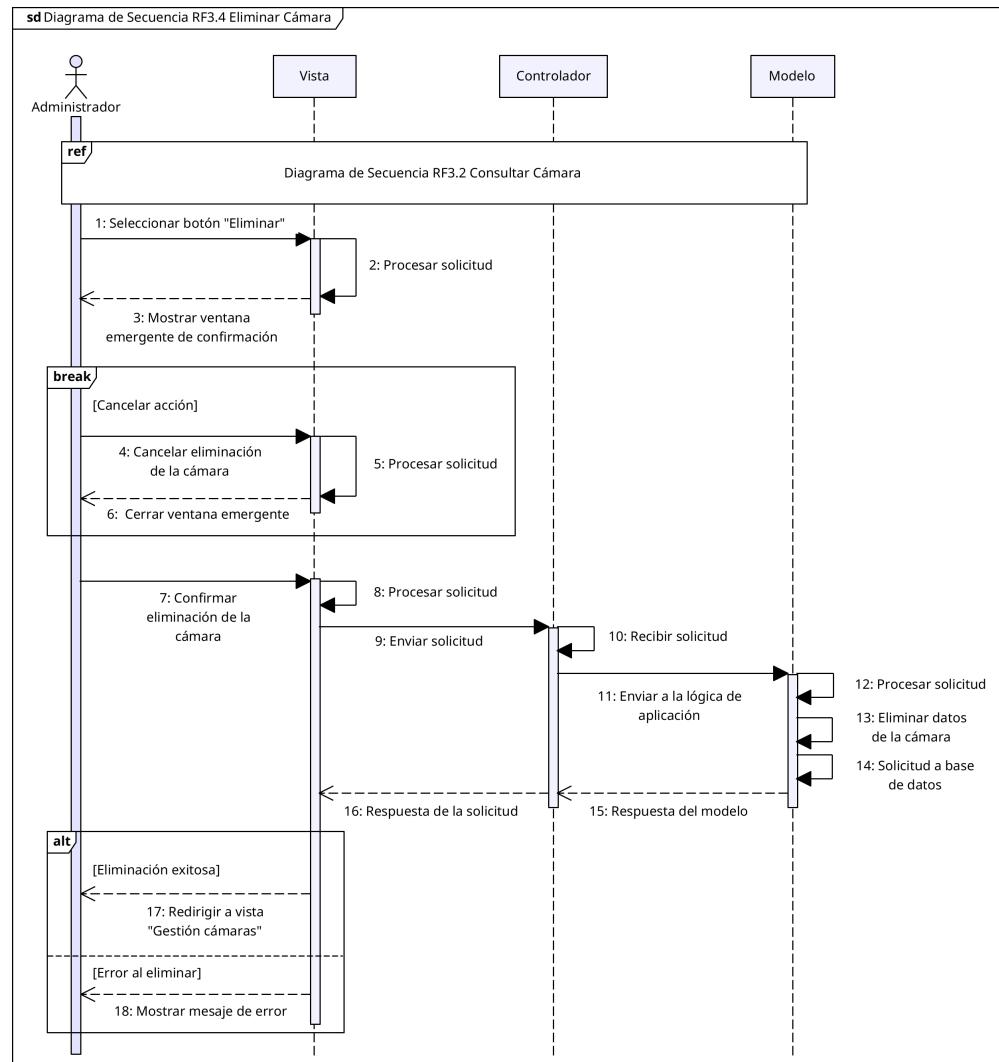
**Figura 21**

Diagrama de Secuencia para Editar Cámara (RF3.3).



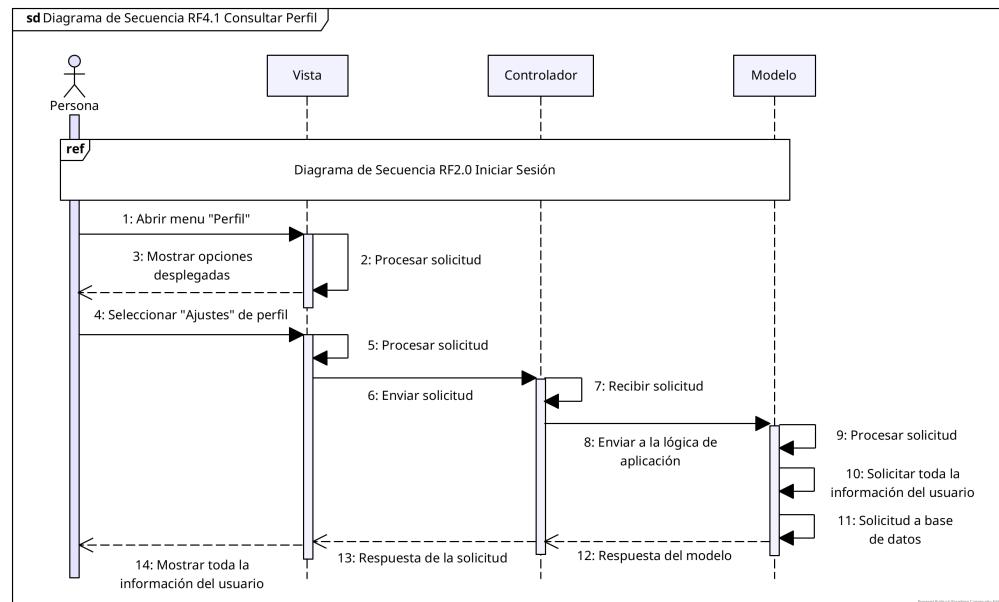
**Figura 22**

Diagrama de Secuencia para Eliminar Cámara (RF3.4).



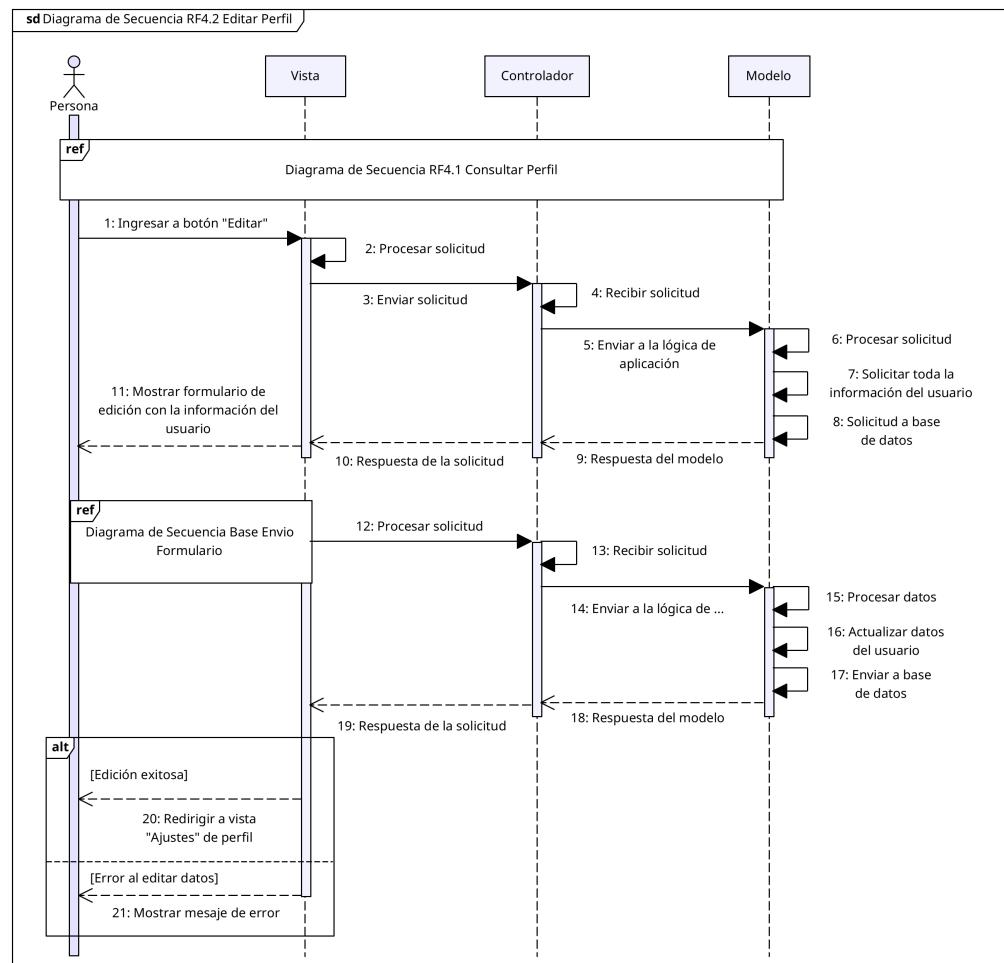
**Figura 23**

*Diagrama de Secuencia para Consultar Perfil (RF4.1).*



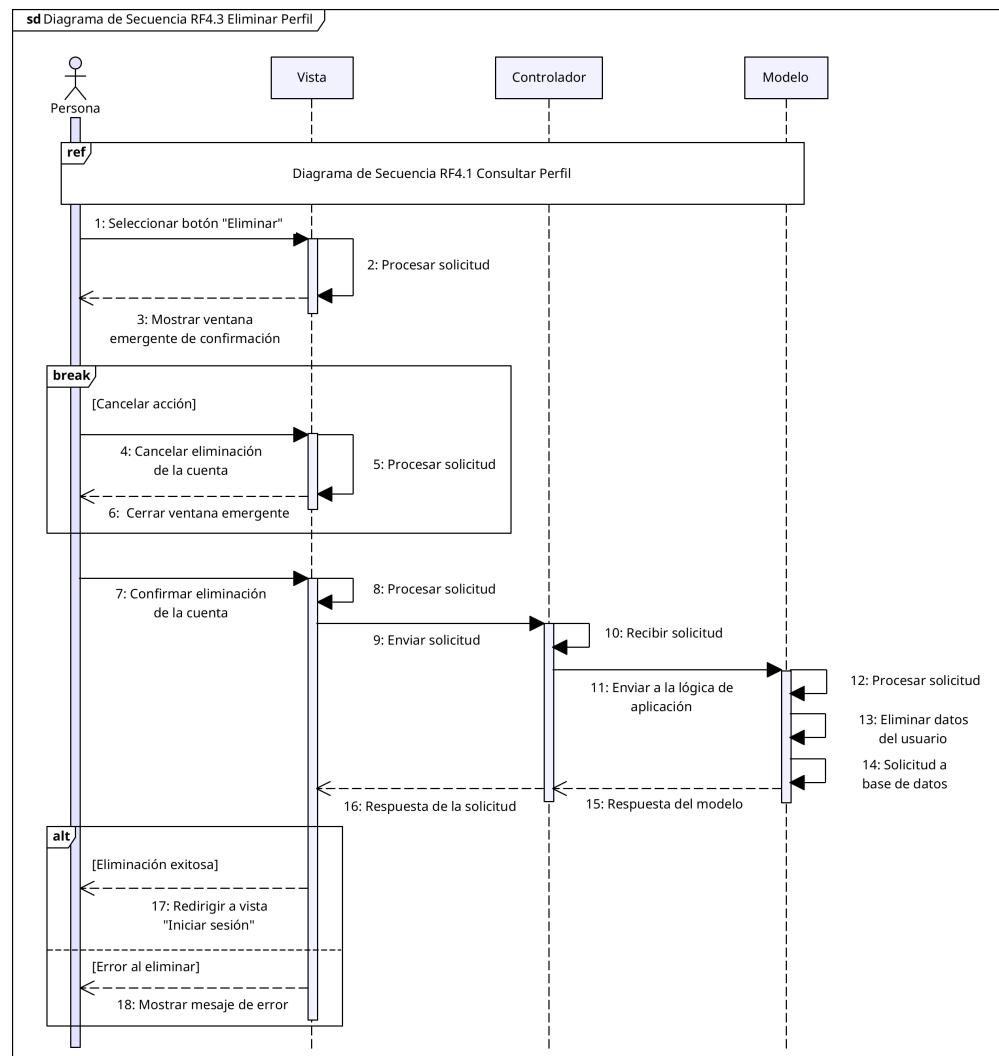
**Figura 24**

Diagrama de Secuencia para Editar Perfil (RF4.2).



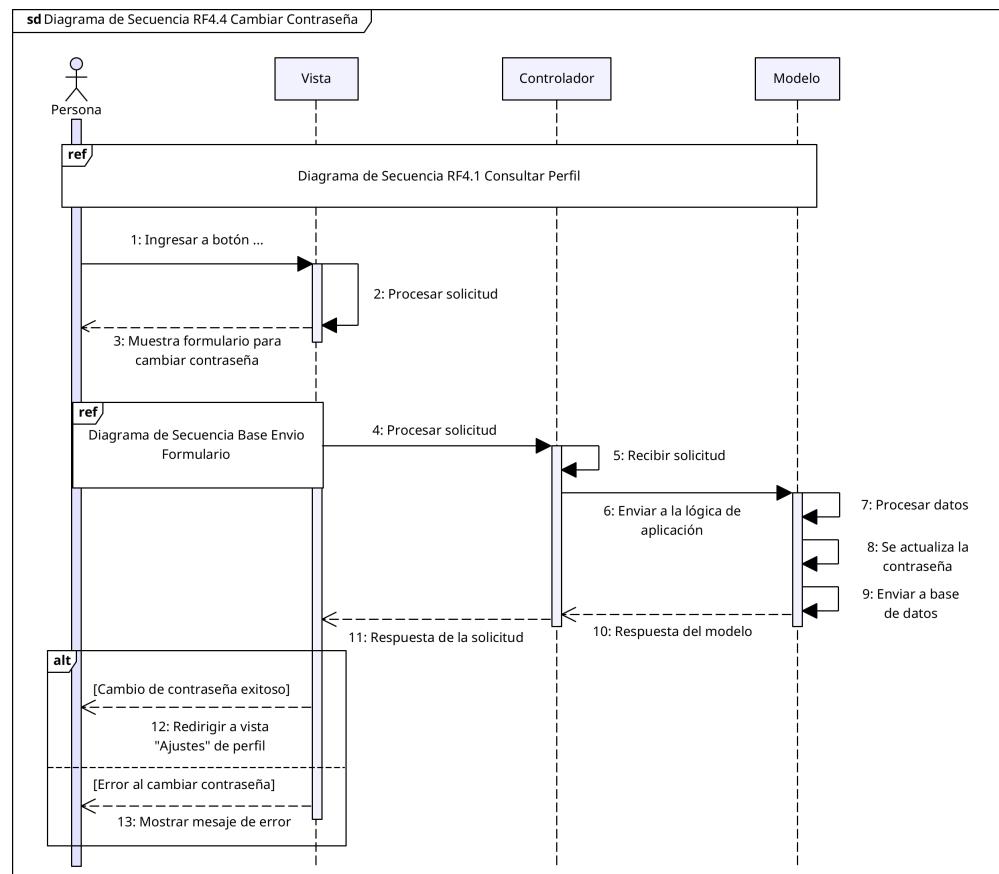
**Figura 25**

Diagrama de Secuencia para Eliminar Perfil (RF4.3).



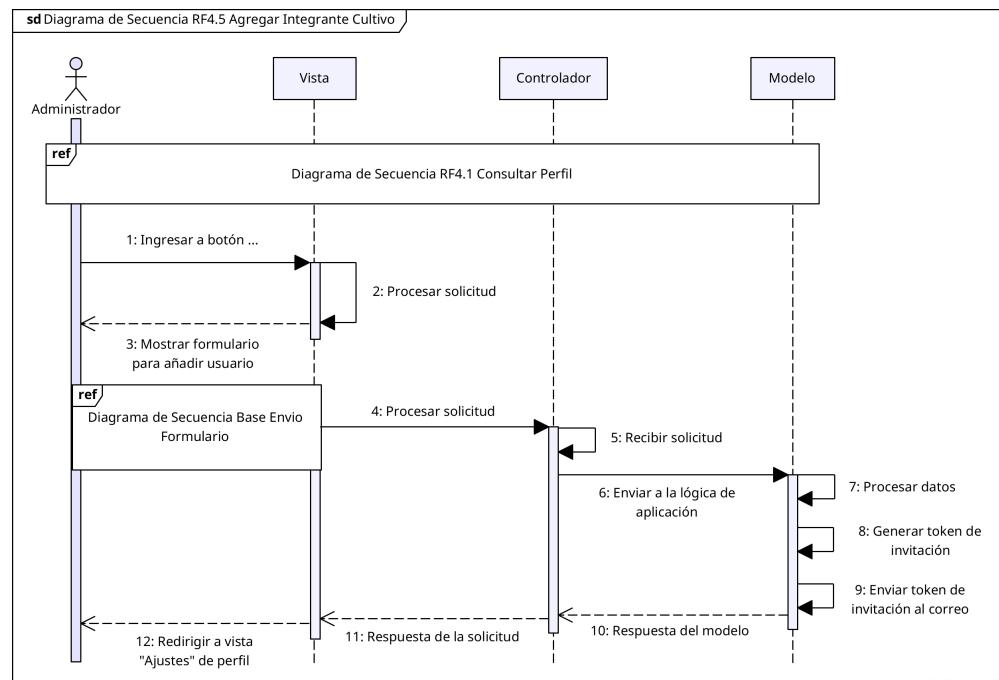
**Figura 26**

*Diagrama de Secuencia para Cambiar Contraseña (RF4.4).*



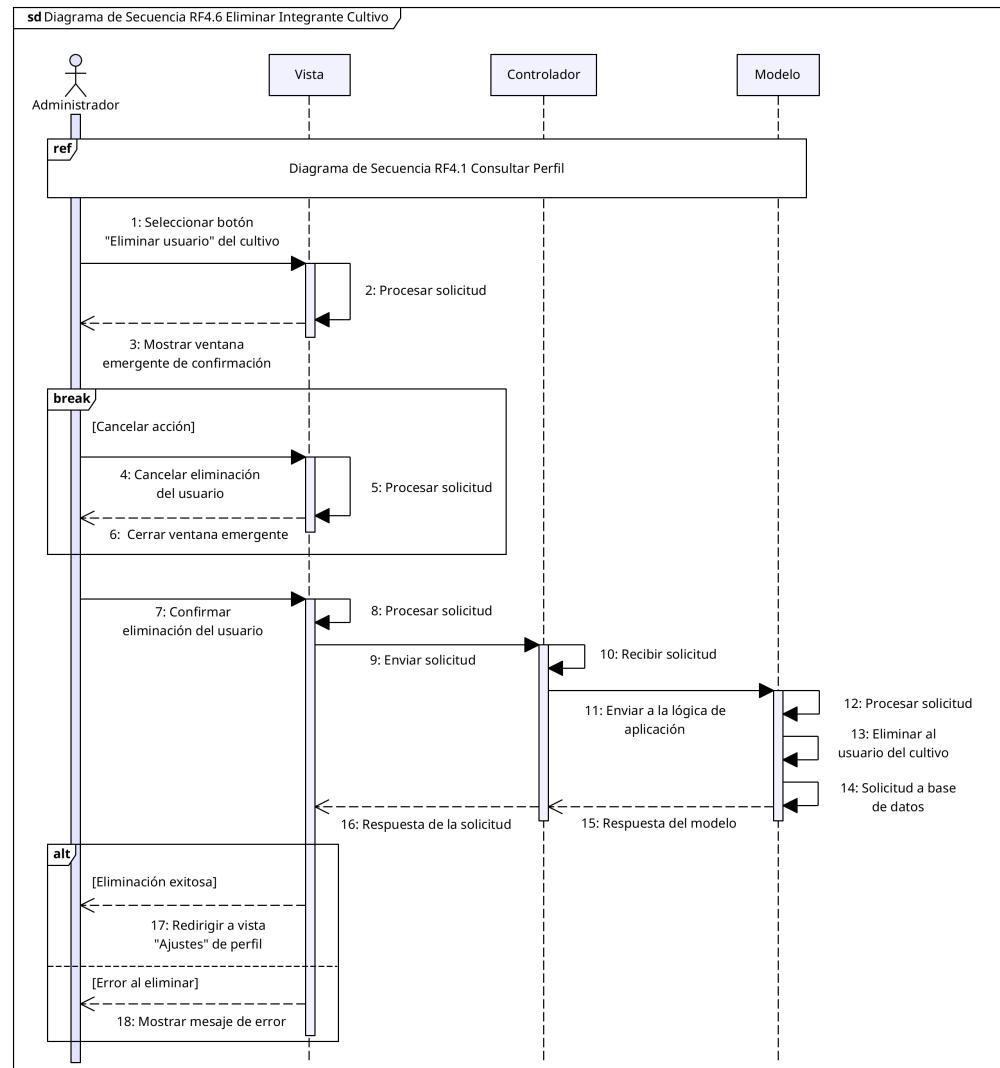
**Figura 27**

Diagrama de Secuencia para Agregar Integrante de Cultivo (RF4.5).



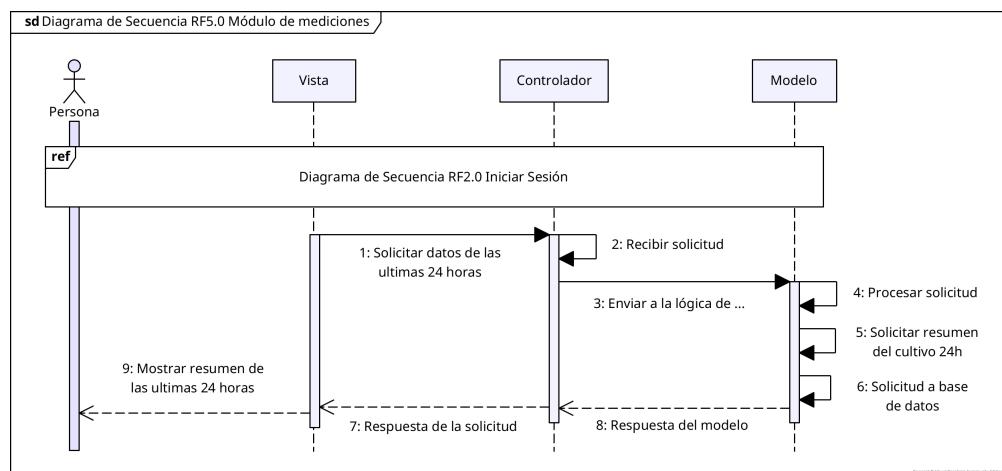
**Figura 28**

Diagrama de Secuencia para Eliminar Integrante de Cultivo (RF4.6).



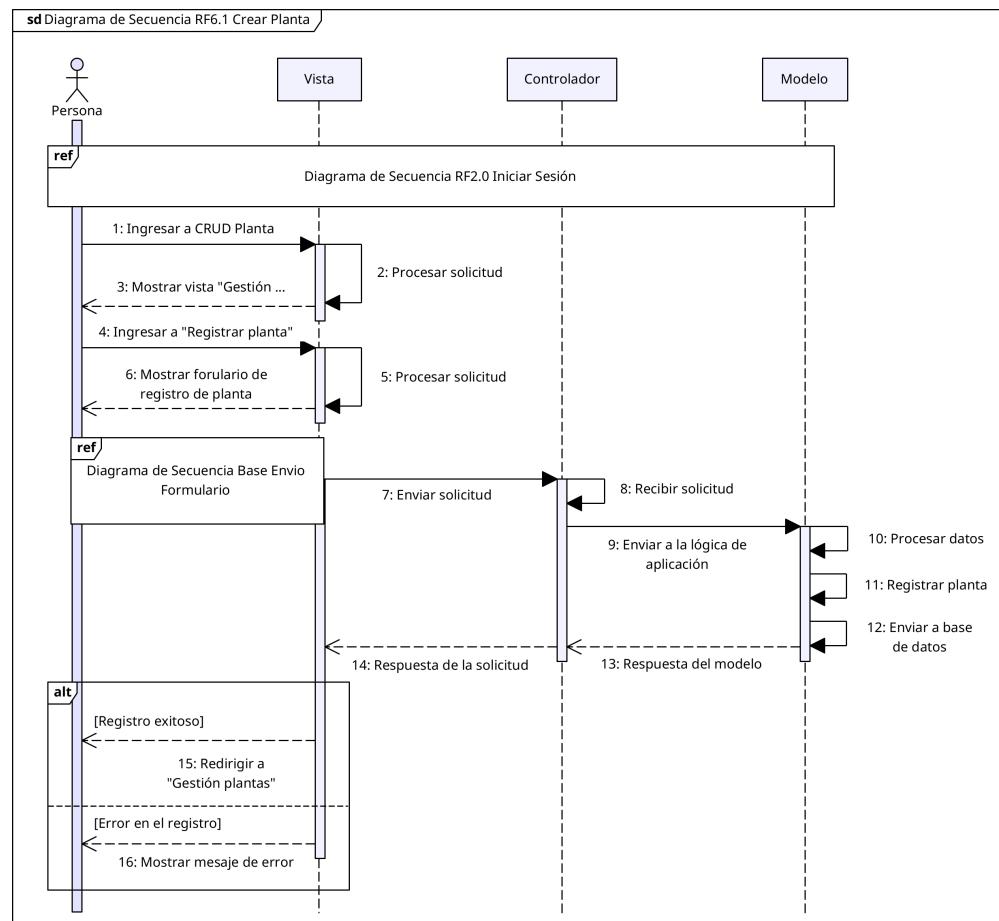
**Figura 29**

Diagrama de Secuencia para el Módulo de Mediciones (RF5.0).



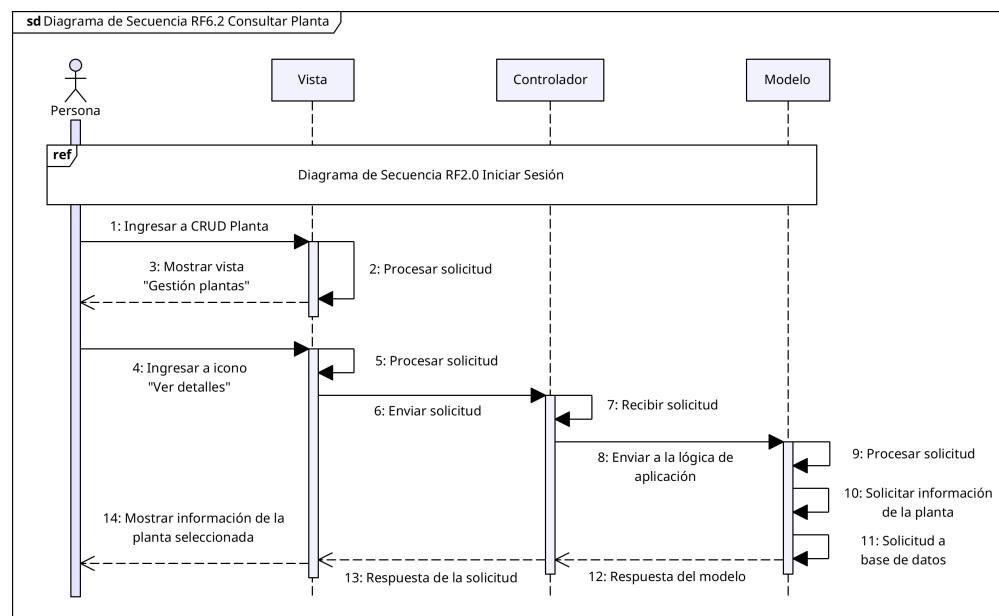
**Figura 30**

Diagrama de Secuencia para Crear Planta (RF6.1).



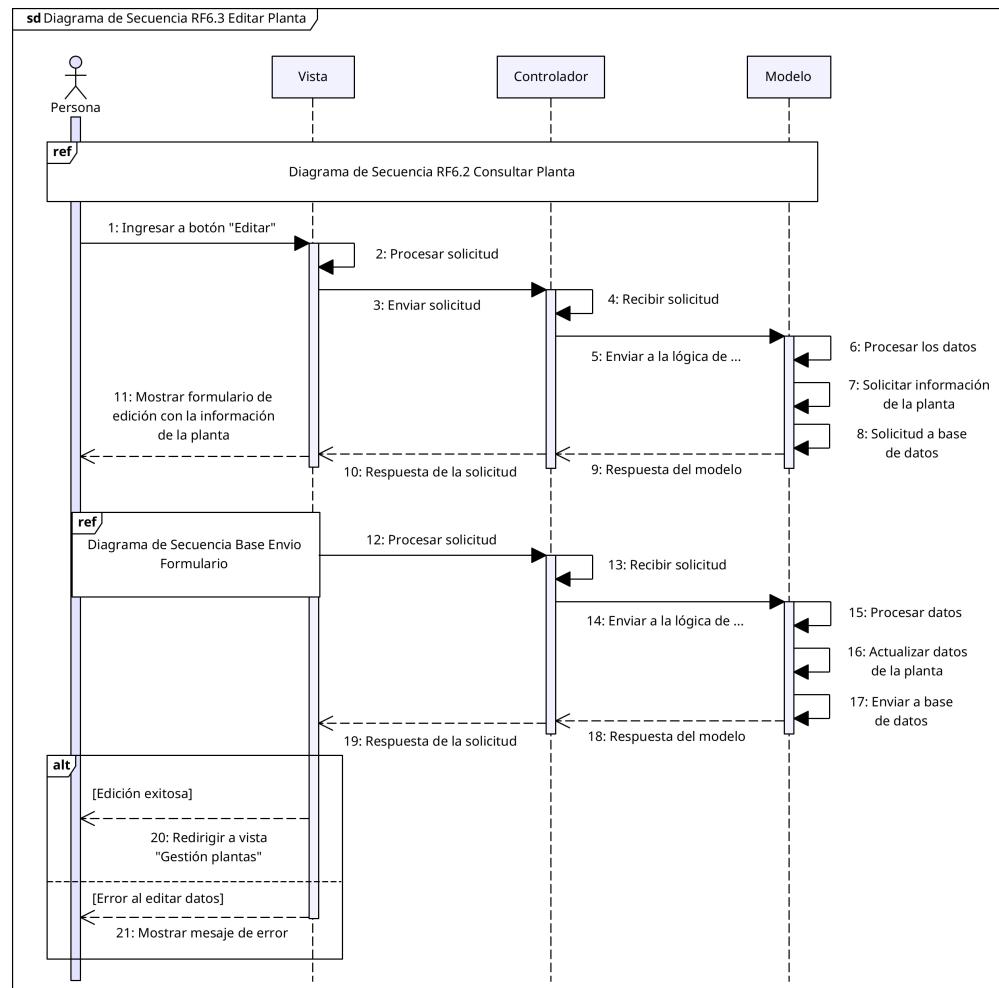
**Figura 31**

*Diagrama de Secuencia para Consultar Planta (RF6.2).*



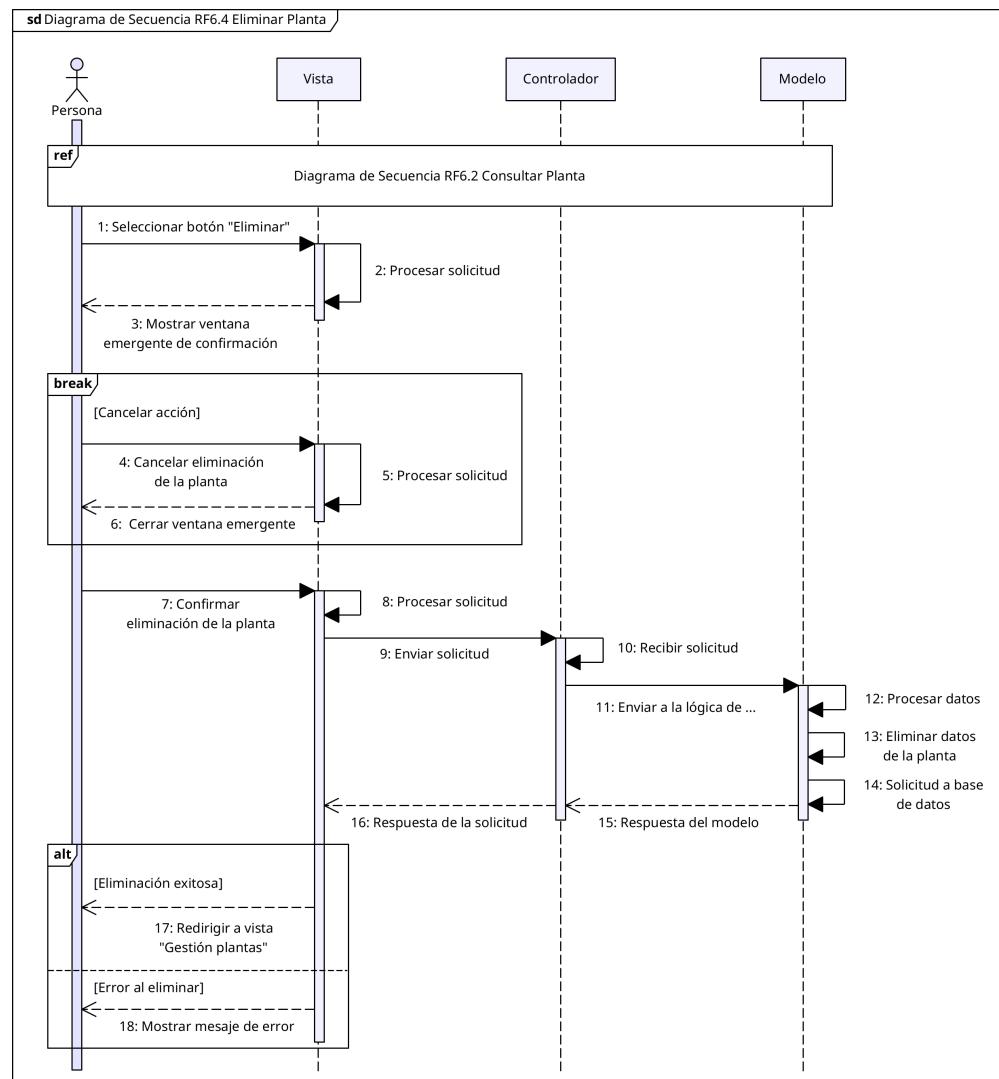
**Figura 32**

Diagrama de Secuencia para Editar Planta (RF6.3).



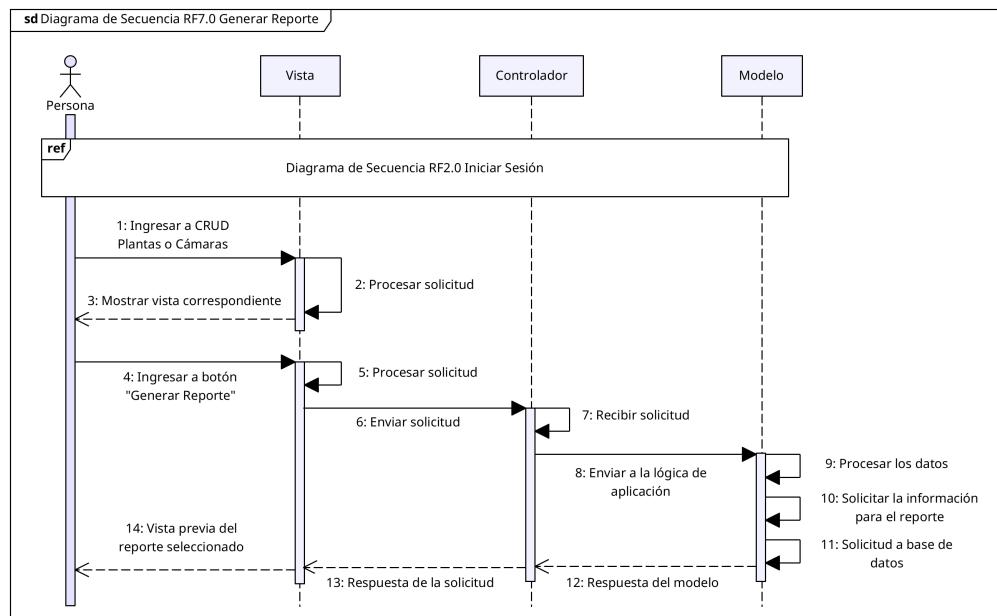
**Figura 33**

*Diagrama de Secuencia para Eliminar Planta (RF6.4).*



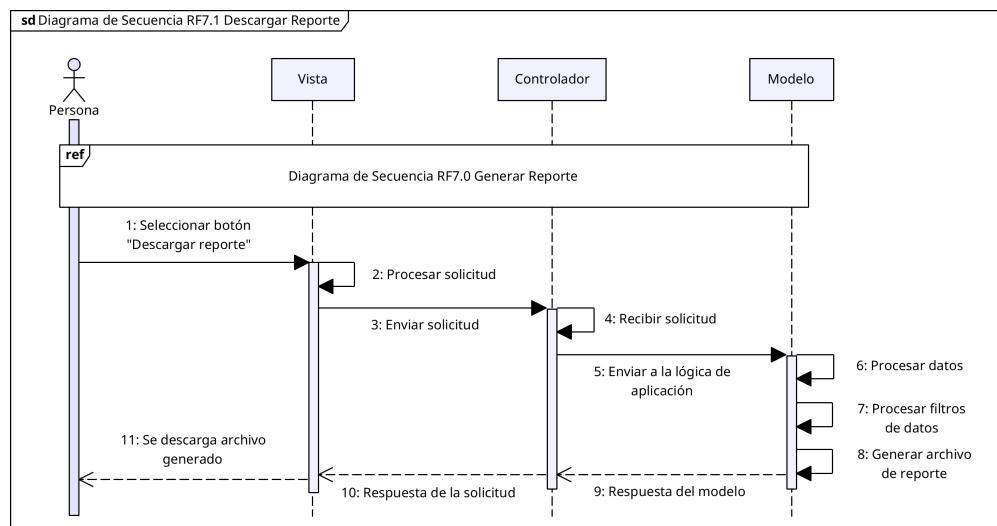
**Figura 34**

Diagrama de Secuencia para Generar Reporte (RF7.0).



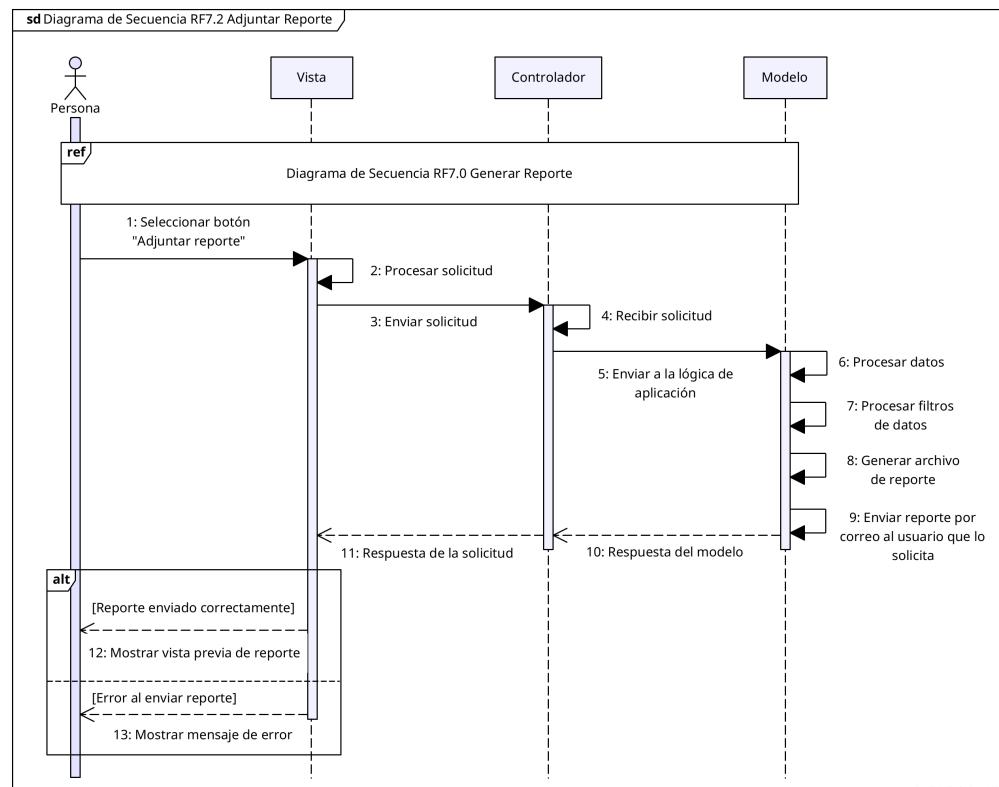
**Figura 35**

Diagrama de Secuencia para Descargar Reporte (RF7.1).



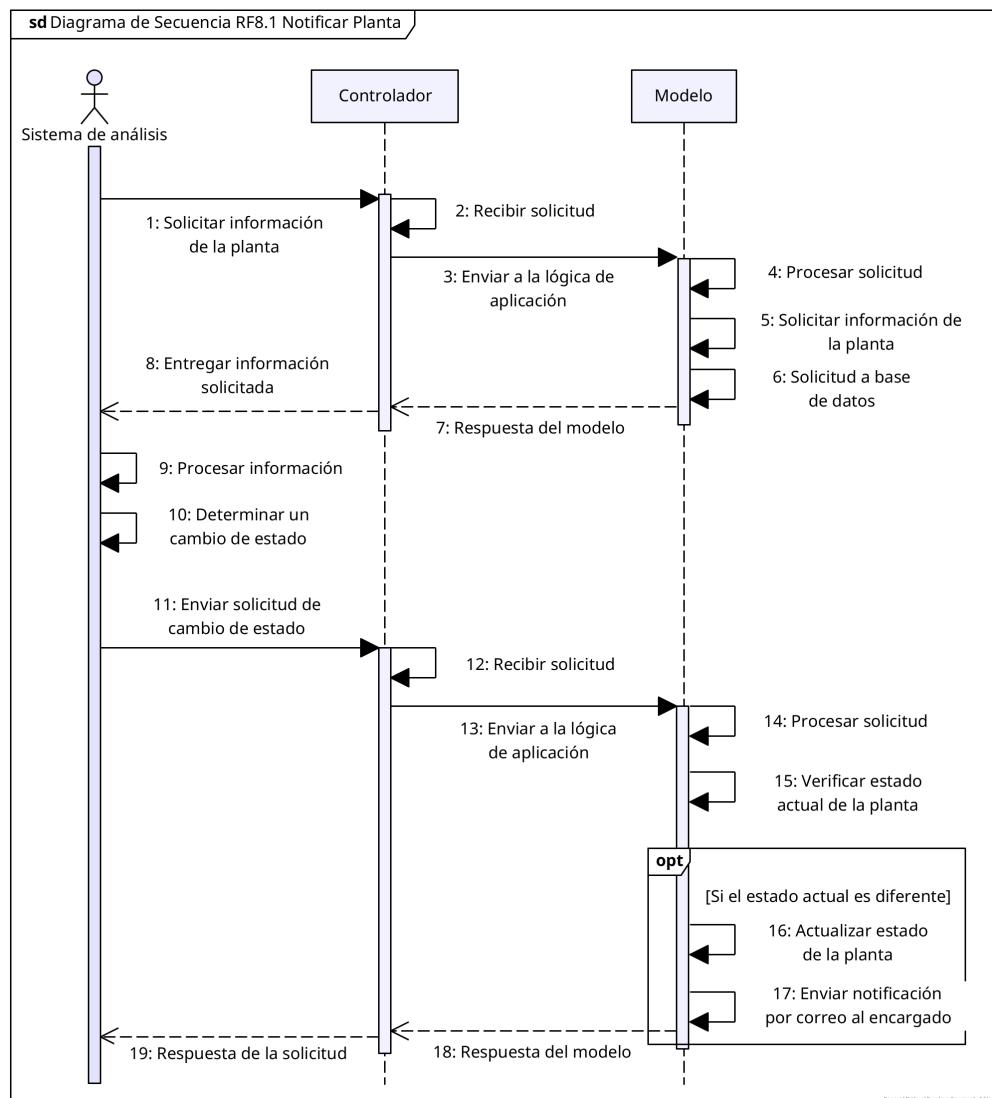
**Figura 36**

Diagrama de Secuencia para Adjuntar Reporte (RF7.2).



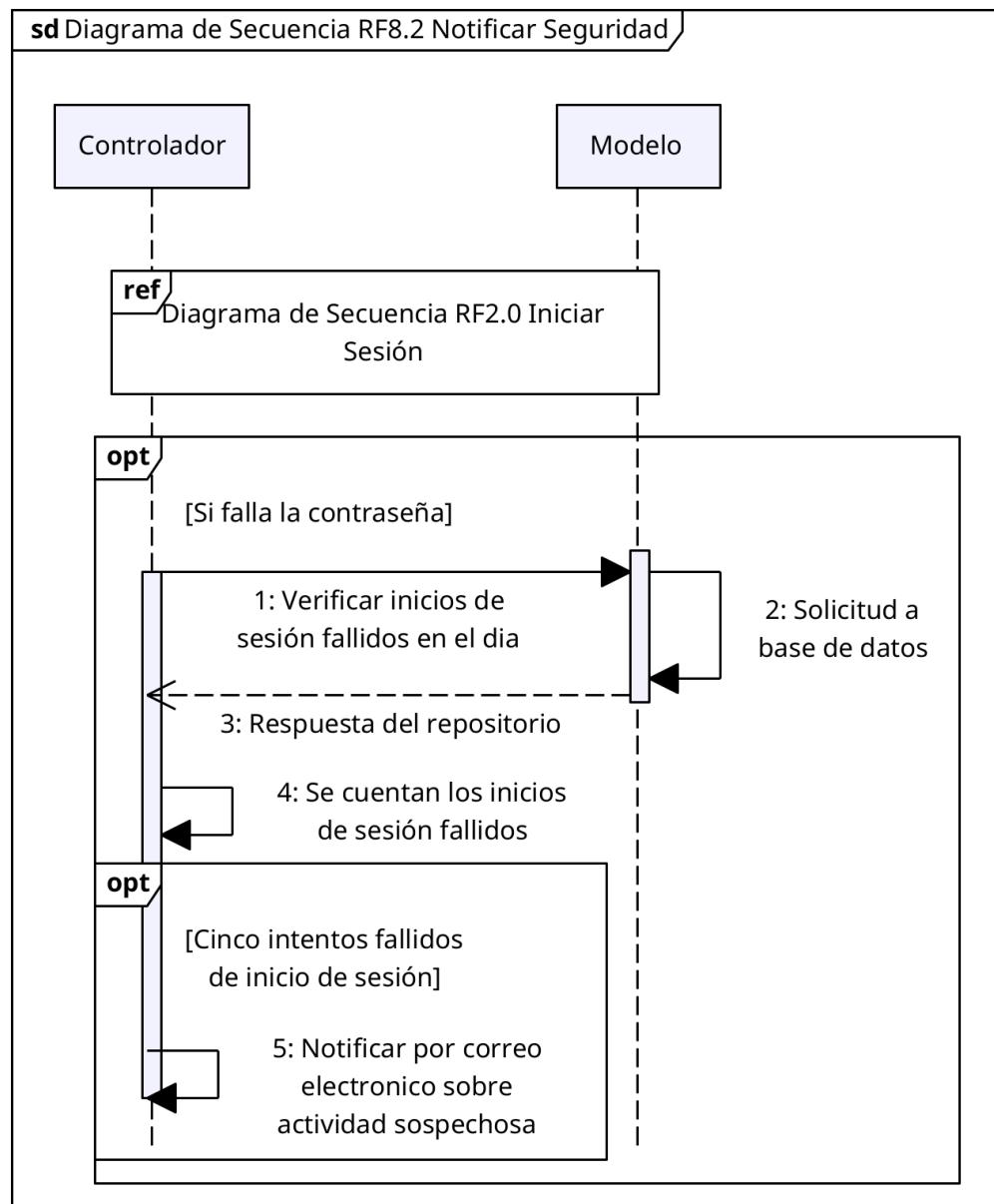
**Figura 37**

Diagrama de Secuencia para Notificar Planta (RF8.1).



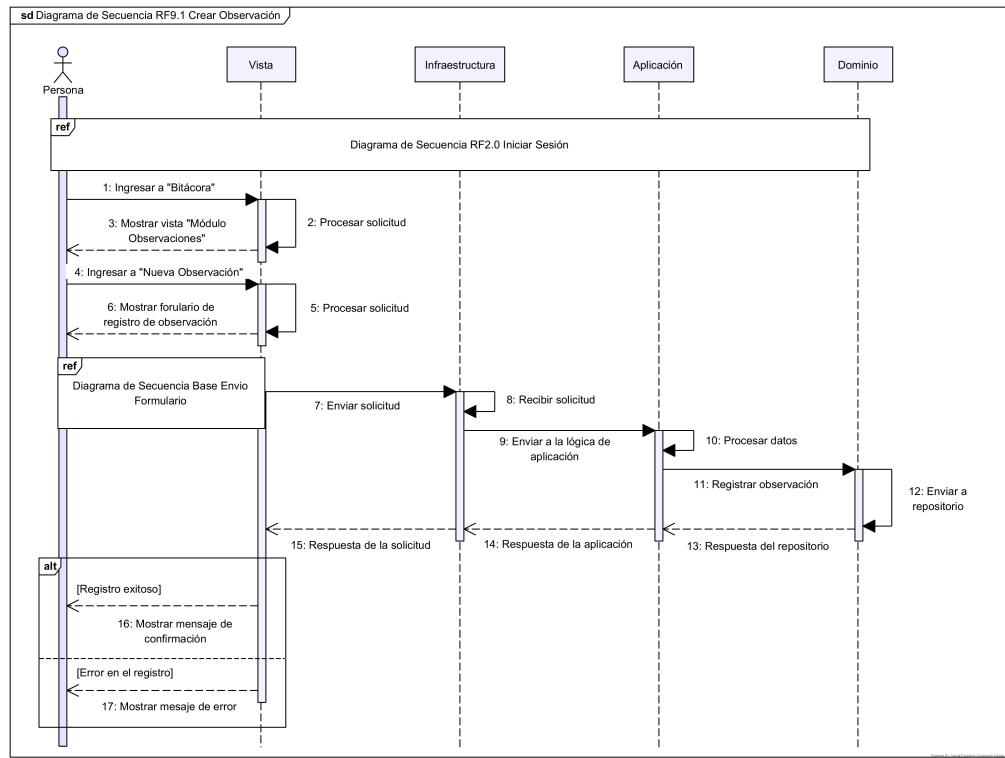
**Figura 38**

Diagrama de Secuencia para Notificar Seguridad (RF8.2).



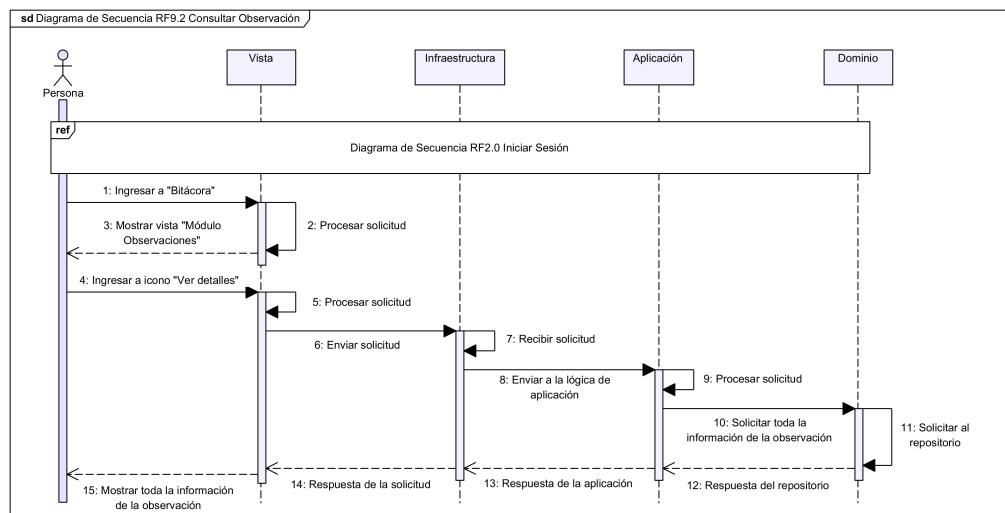
**Figura 39**

Diagrama de Secuencia para Crear Observación (RF9.1).



**Figura 40**

Diagrama de Secuencia para Consultar Observación (RF9.1).



#### **2.4.4. Diagramas de Actividades**

Una actividad muestra el flujo de control entre las actividades computacionales involucradas en la realización de un cálculo o un flujo de trabajo (Rumbaugh y cols., 2007). Una acción es un paso computacional primitivo y un nodo de actividad es un grupo de acciones o subactividades (Rumbaugh y cols., 2007). Una actividad describe tanto el cómputo secuencial como el concurrente (Rumbaugh y cols., 2007).

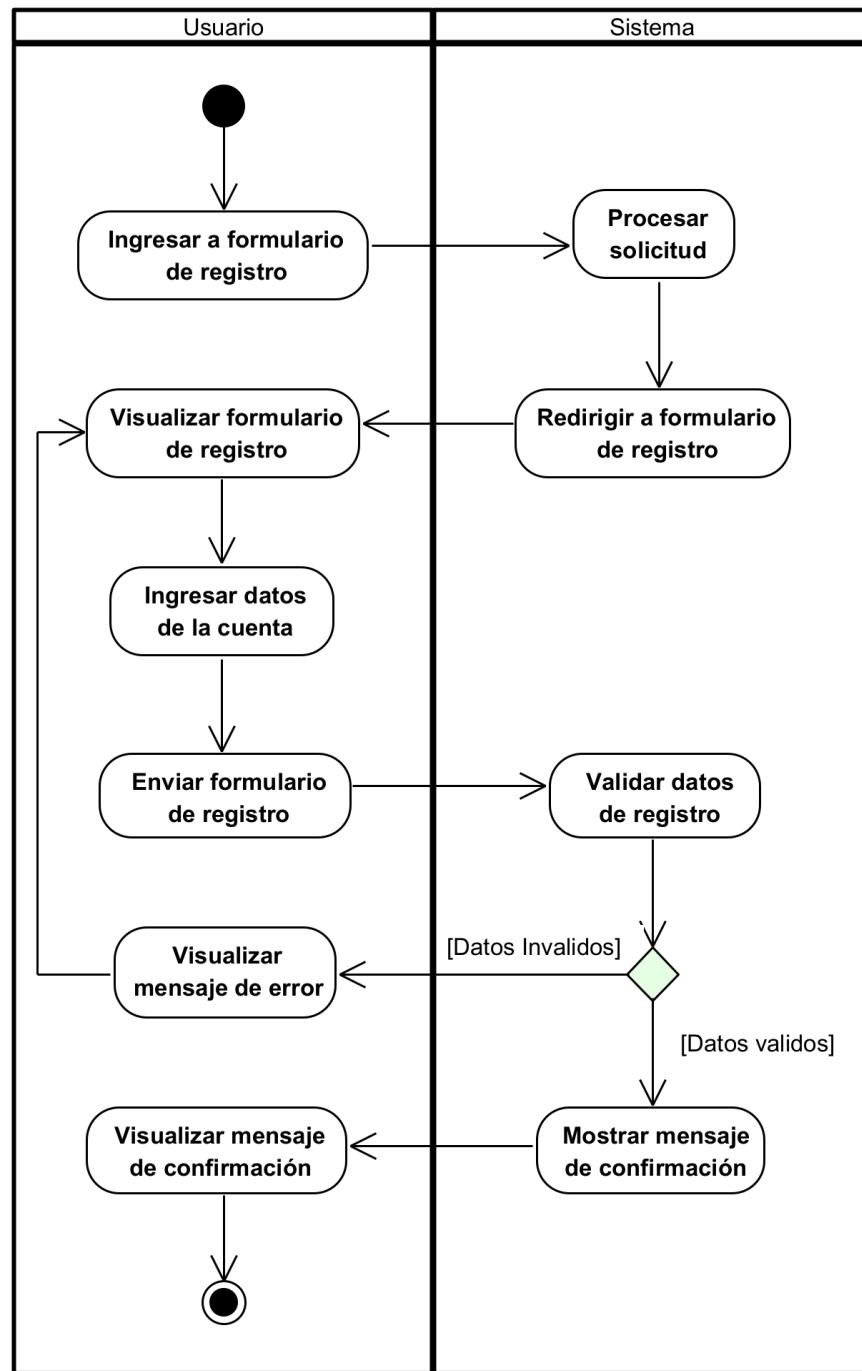
Los diagramas de actividad presentados a continuación modelan los flujos de trabajo (*workflows*) asociados a procesos o funcionalidades clave del sistema. Estos diagramas utilizan particiones verticales, comúnmente conocidas como calles o *swimlanes*, para asignar claramente la responsabilidad de cada acción a un participante específico del proceso. En la mayoría de los diagramas de este documento, se utilizan dos calles principales:

- **Usuario:** Esta calle agrupa todas aquellas actividades que son ejecutadas directamente por la persona que interactúa con la interfaz gráfica del sistema. Representa las acciones del usuario final, ya sea que actúe con el rol de **Usuario** o de **Administrador**. Ejemplos típicos de actividades en esta calle incluyen: ingresar datos en un formulario, seleccionar opciones, iniciar una acción (como presionar un botón) y visualizar mensajes o resultados mostrados por la aplicación.
- **Sistema:** Esta calle engloba todas las actividades y procesos que son realizados internamente por la aplicación software (backend y/o frontend). Representa las operaciones automáticas del sistema, tales como: procesar la información enviada por el usuario, validar datos según reglas de negocio, ejecutar algoritmos, consultar o actualizar información en la base de datos, determinar estados, generar respuestas y preparar la información que se devolverá a la interfaz para ser visualizada por el usuario.

La separación de actividades entre estas dos calles principales (*Usuario* y *Sistema*) permite visualizar de manera clara la interacción entre el usuario humano y la lógica interna de la aplicación a lo largo de un flujo de trabajo específico. Otros diagramas podrían incluir calles adicionales si participan otros actores o sistemas externos específicos en un proceso particular.

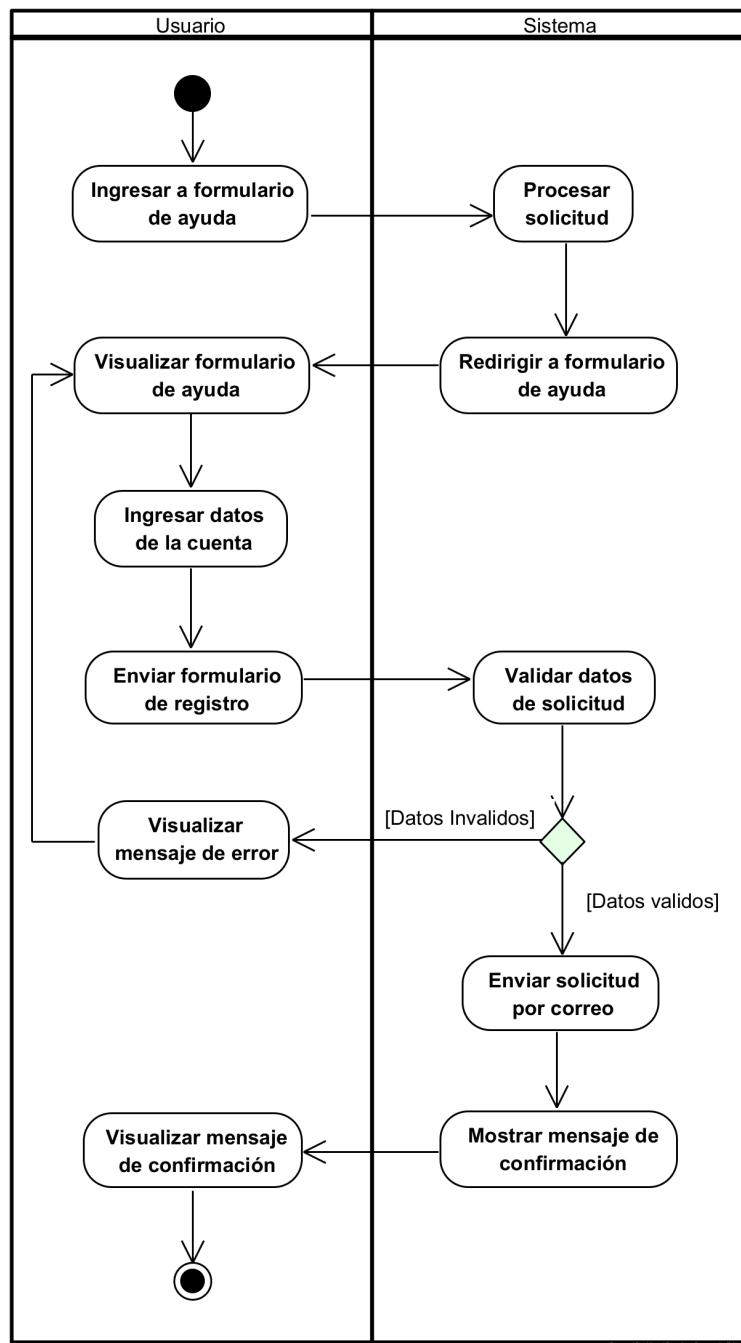
**Figura 41**

Diagrama de Actividad para el Registro (RF1.0).



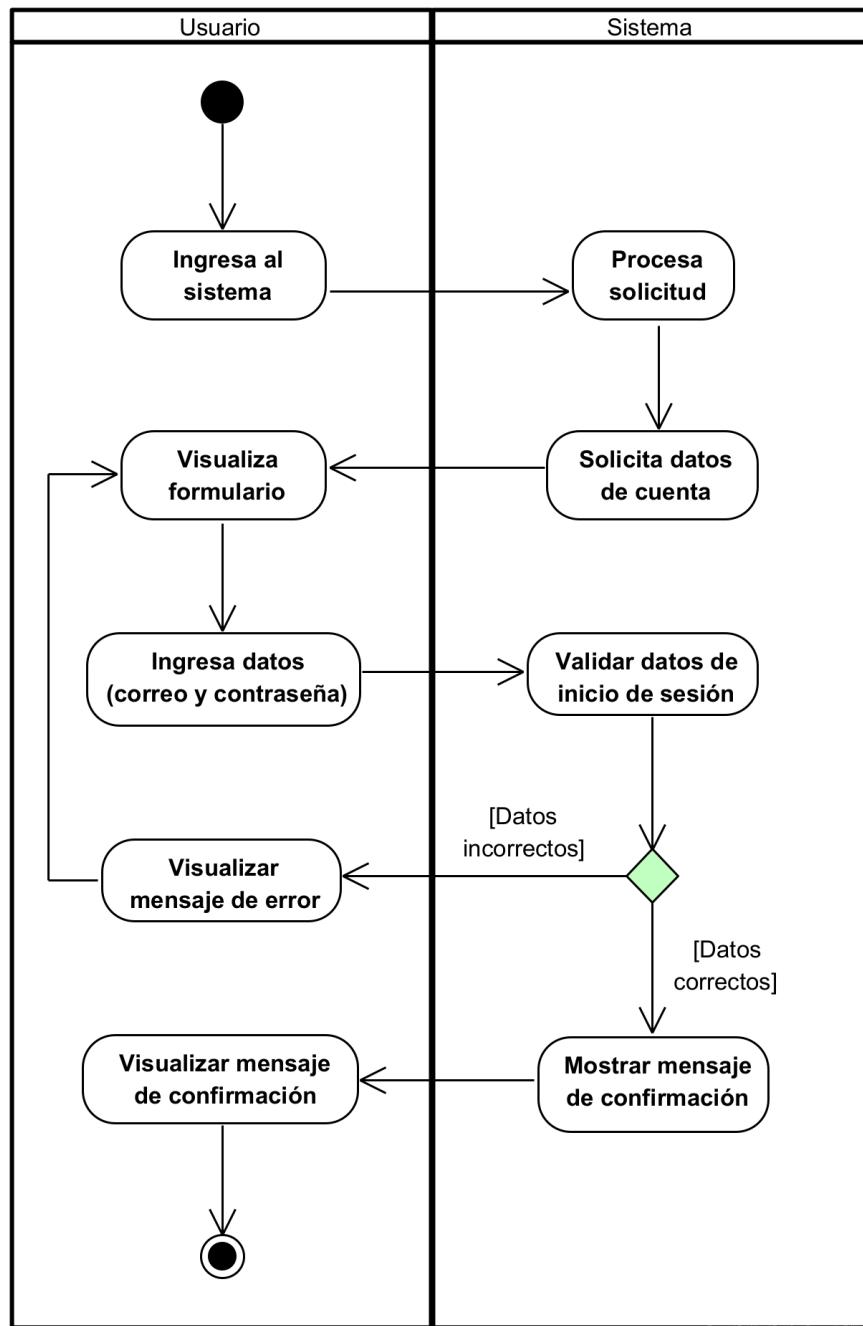
**Figura 42**

Diagrama de Actividad para Solicitar Código (RF1.1).



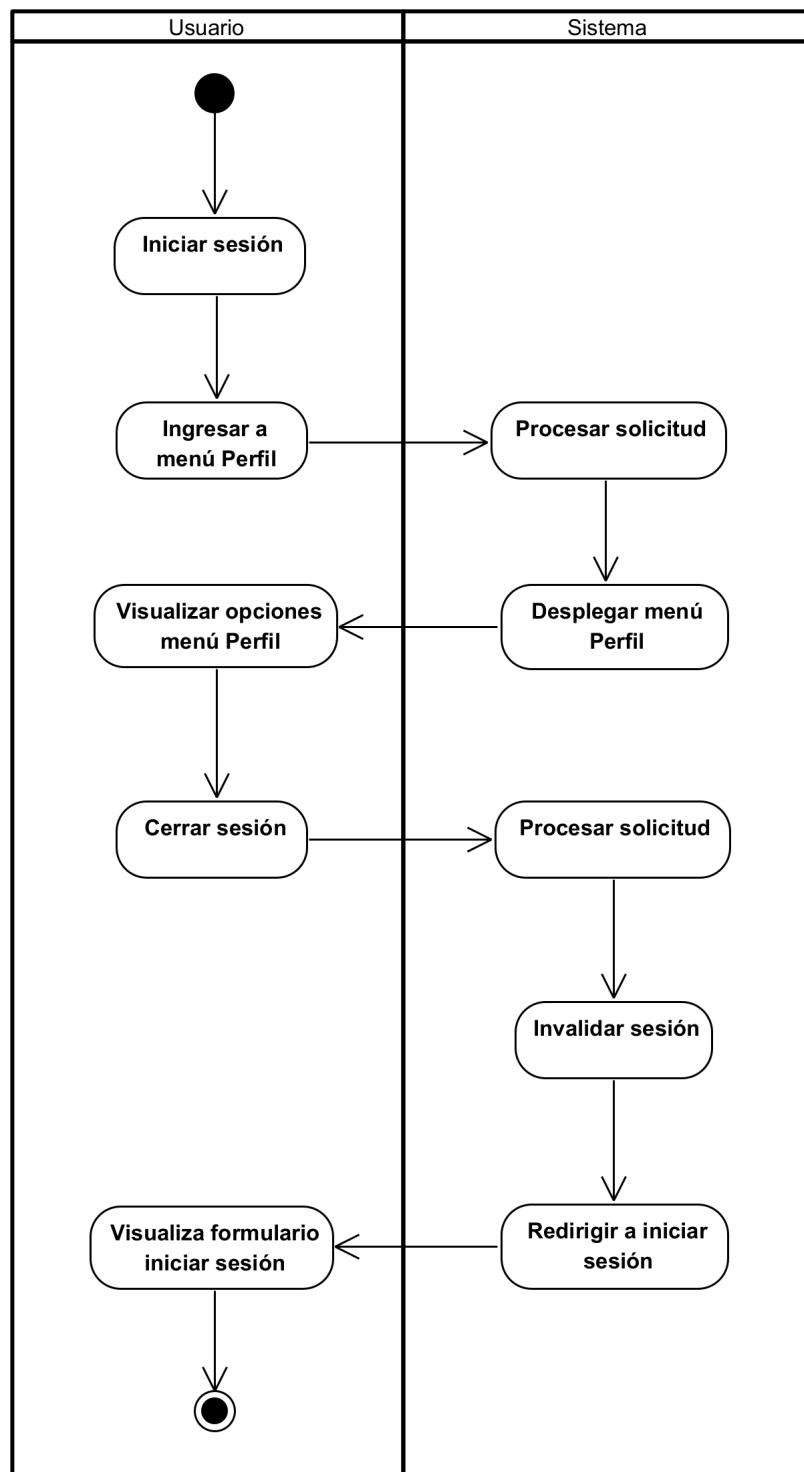
**Figura 43**

Diagrama de Actividad para Iniciar Sesión (RF2.0).



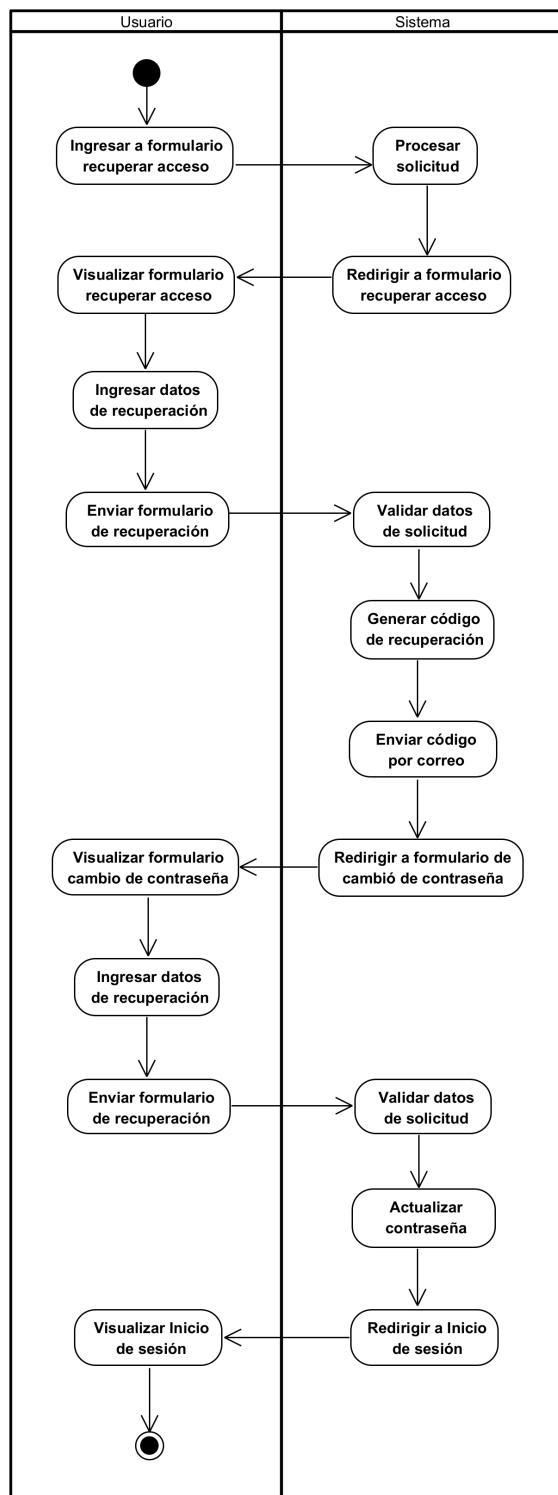
**Figura 44**

Diagrama de Actividad para Cerrar Sesión (RF2.1).



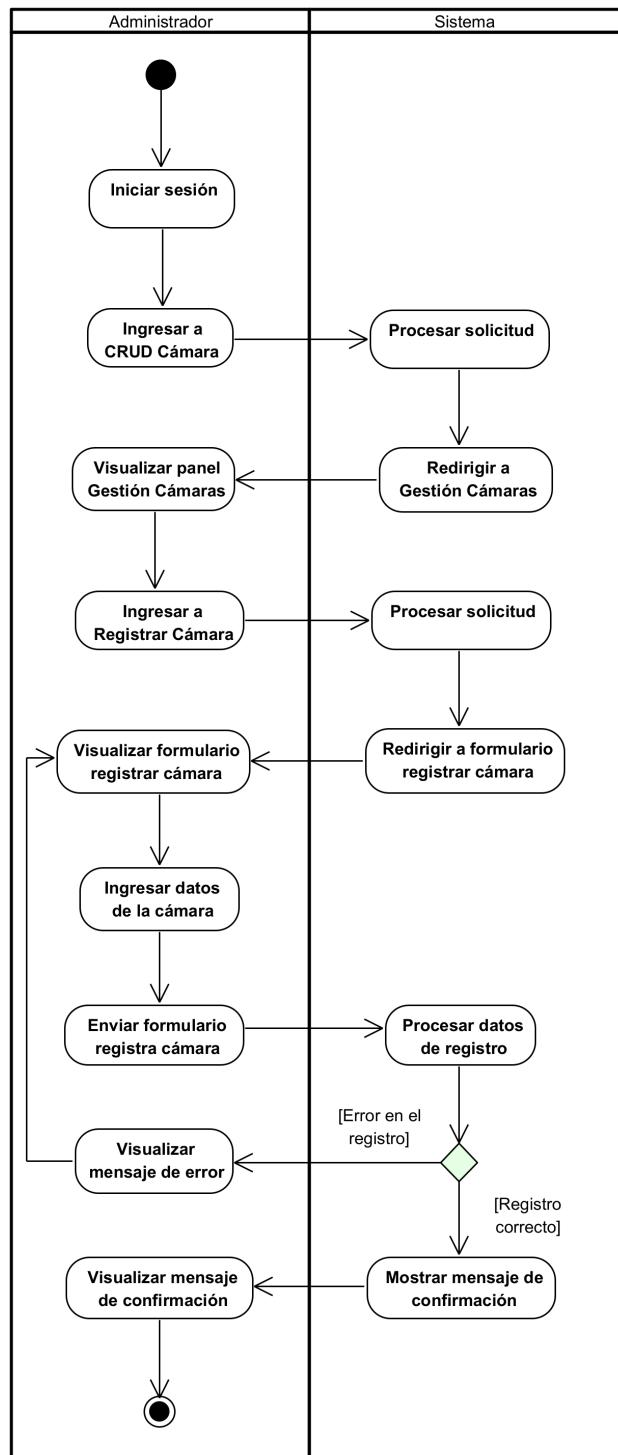
**Figura 45**

*Diagrama de Actividad para Recuperar Contraseña (RF2.2).*



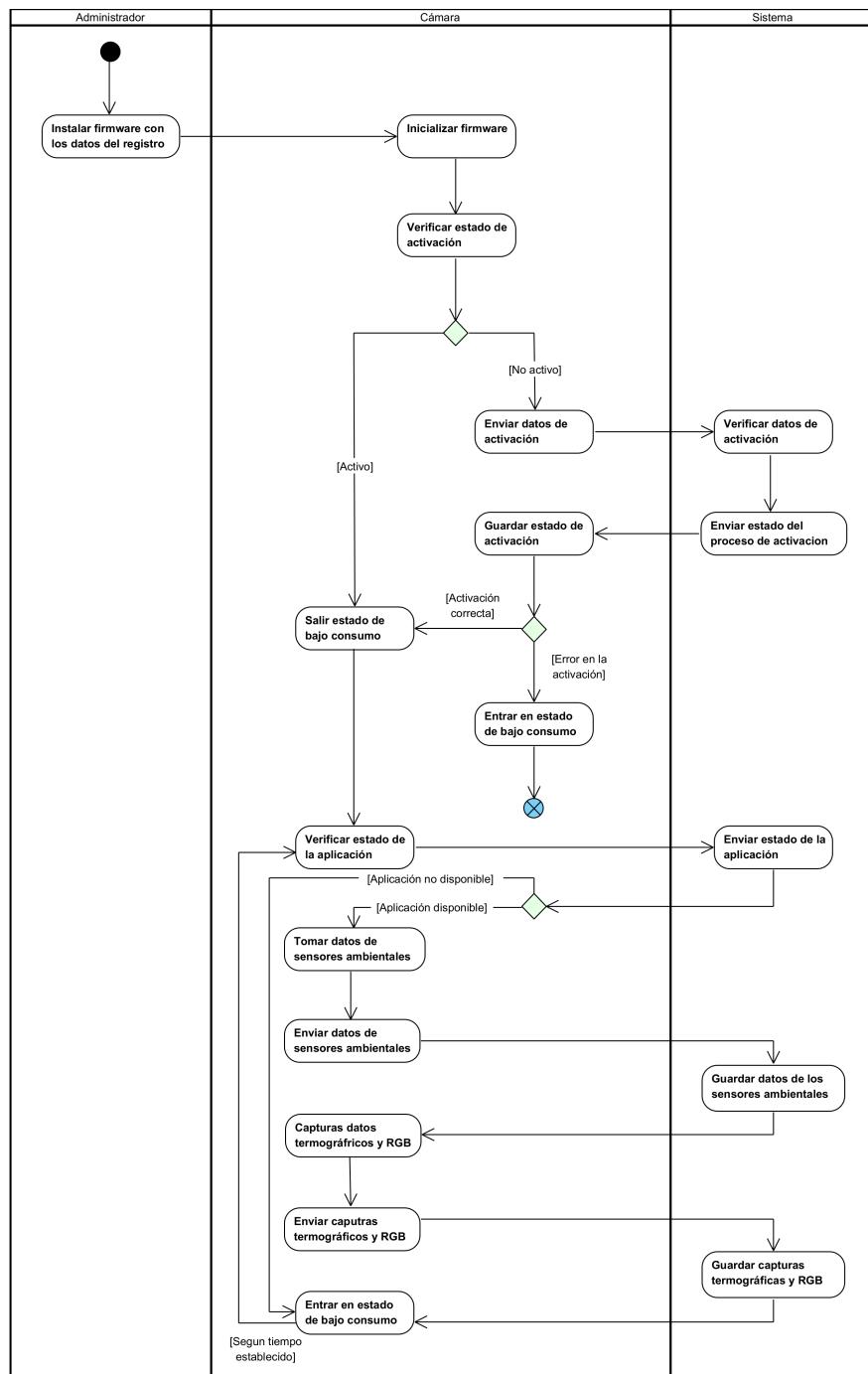
**Figura 46**

Diagrama de Actividad para Crear Cámara (RF3.1).



**Figura 47**

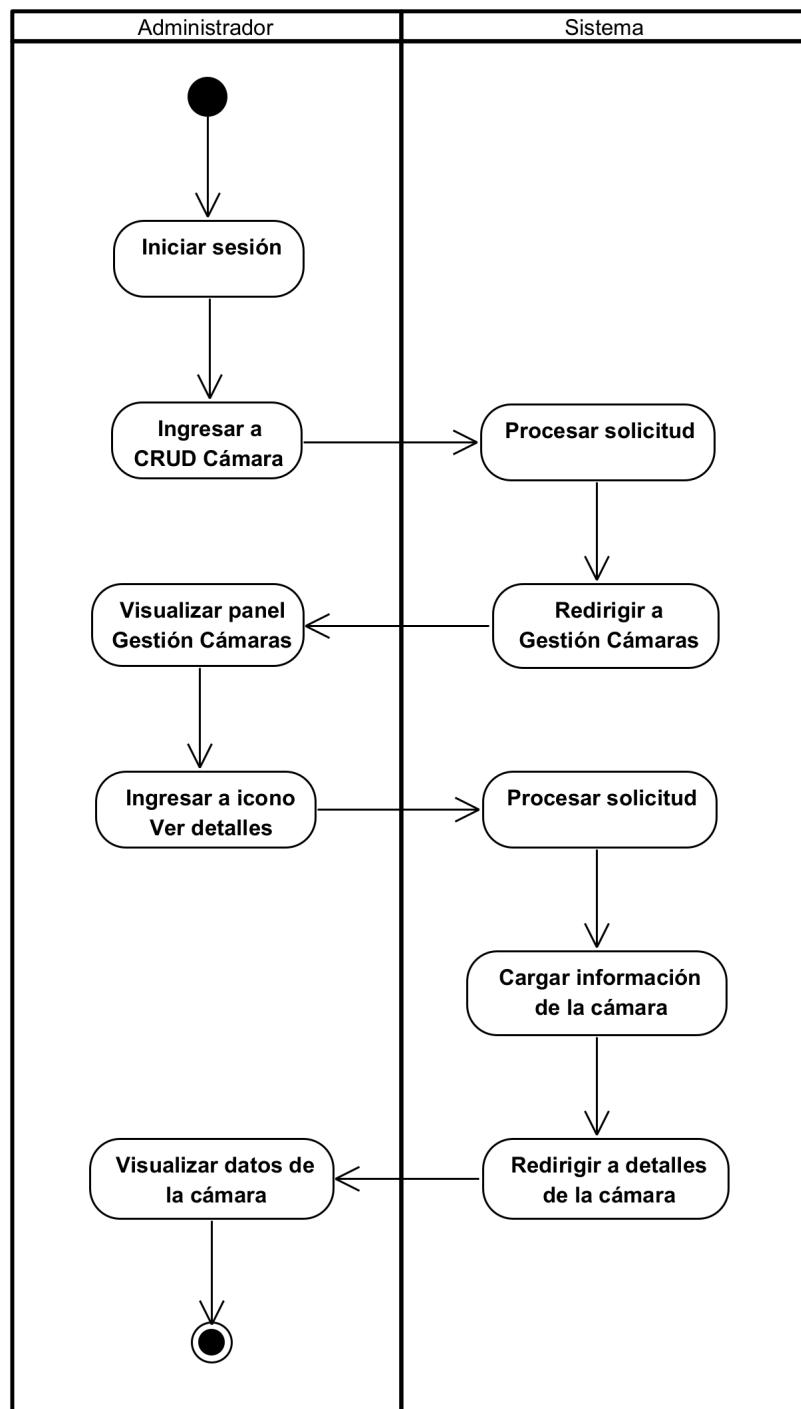
Diagrama de Actividad para Activar Cámara (RF3.1.1).



El diagrama de actividad de la Figura 47 muestra el flujo de trabajo para la activación e inicio de operación del dispositivo físico. Conforme a la descripción general proporcionada al inicio de esta sección, las calles **Administrador** y **Sistema** representan las acciones del usuario y del backend, respectivamente. La calle **Hardware**, corresponden a la lógica ejecutada por el firmware del propio dispositivo. Este flujo describe cómo el hardware gestiona su activación inicial y luego opera en un ciclo de verificación, recolección, envío y espera, interactuando con el **Sistema** cuando es necesario y gestionando su consumo de energía.

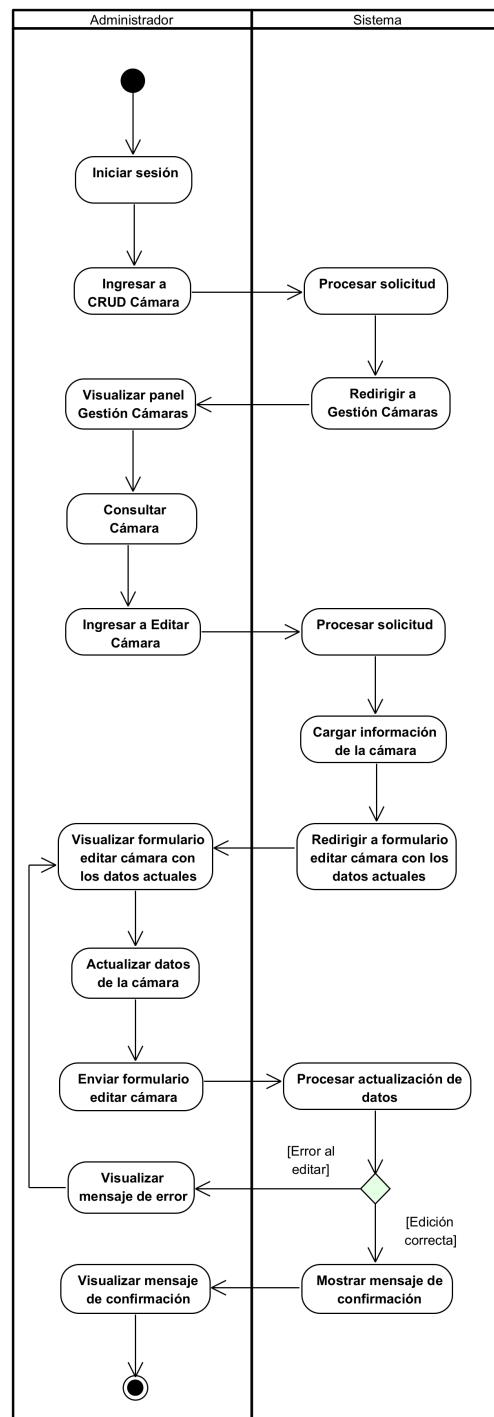
**Figura 48**

Diagrama de Actividad para Consultar Cámara (RF3.2).



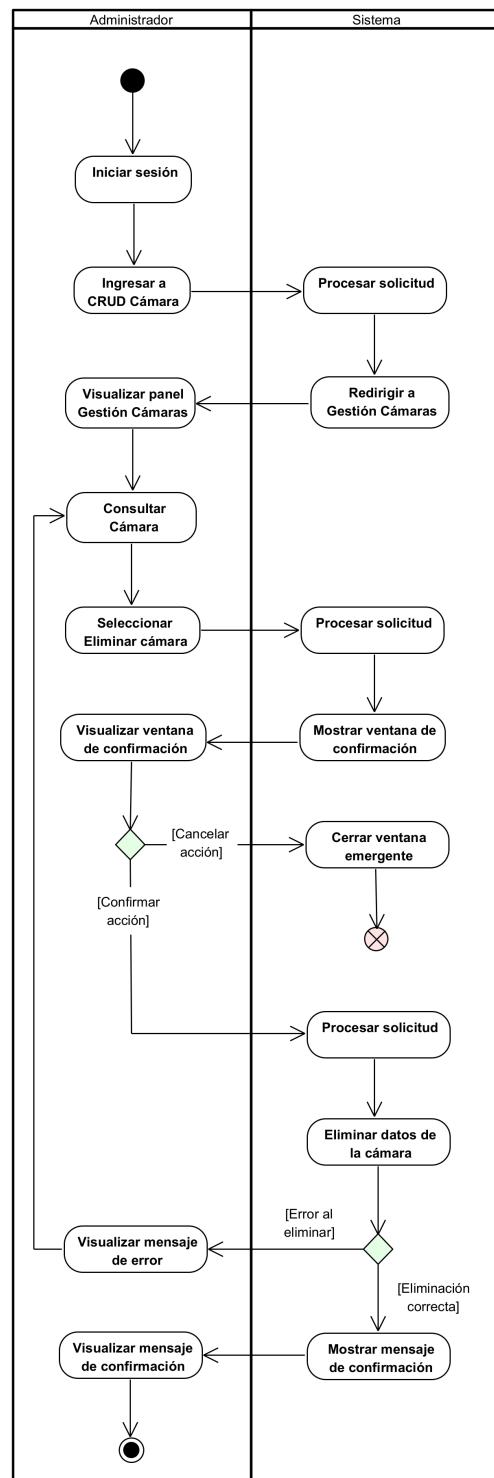
**Figura 49**

Diagrama de Actividad para Editar Cámara (RF3.3).



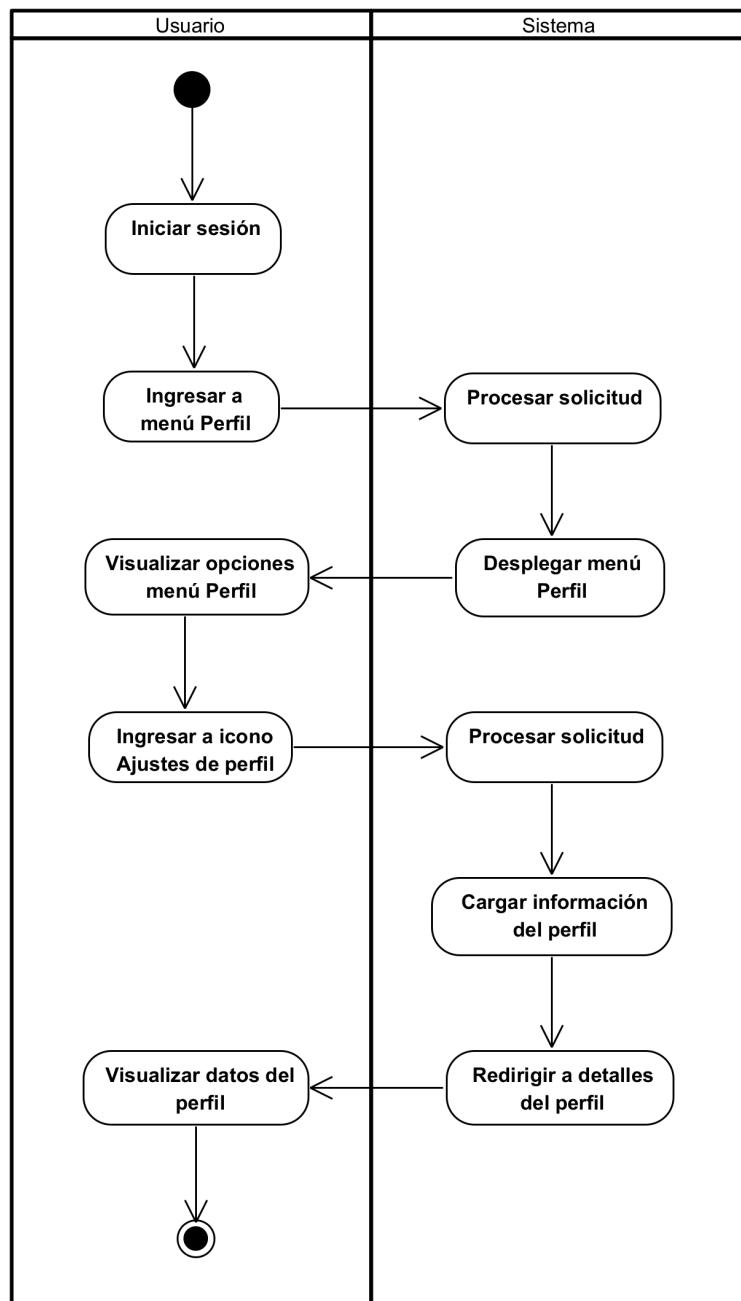
**Figura 50**

Diagrama de Actividad para Eliminar Cámara (RF3.4).



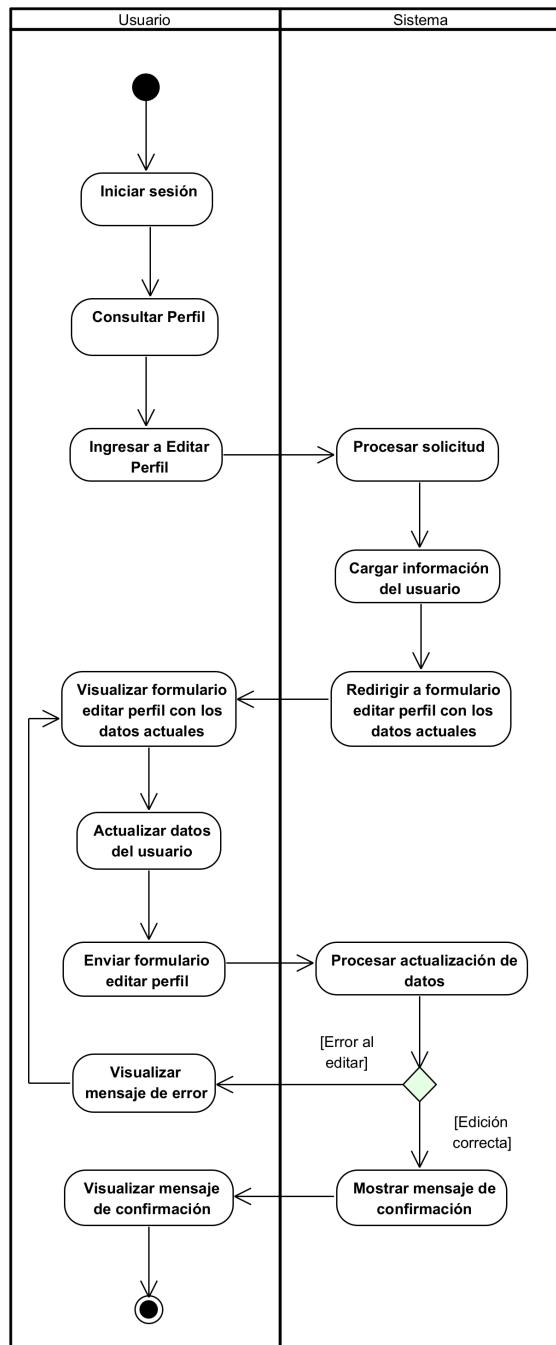
**Figura 51**

*Diagrama de Actividad para Consultar Perfil (RF4.1).*



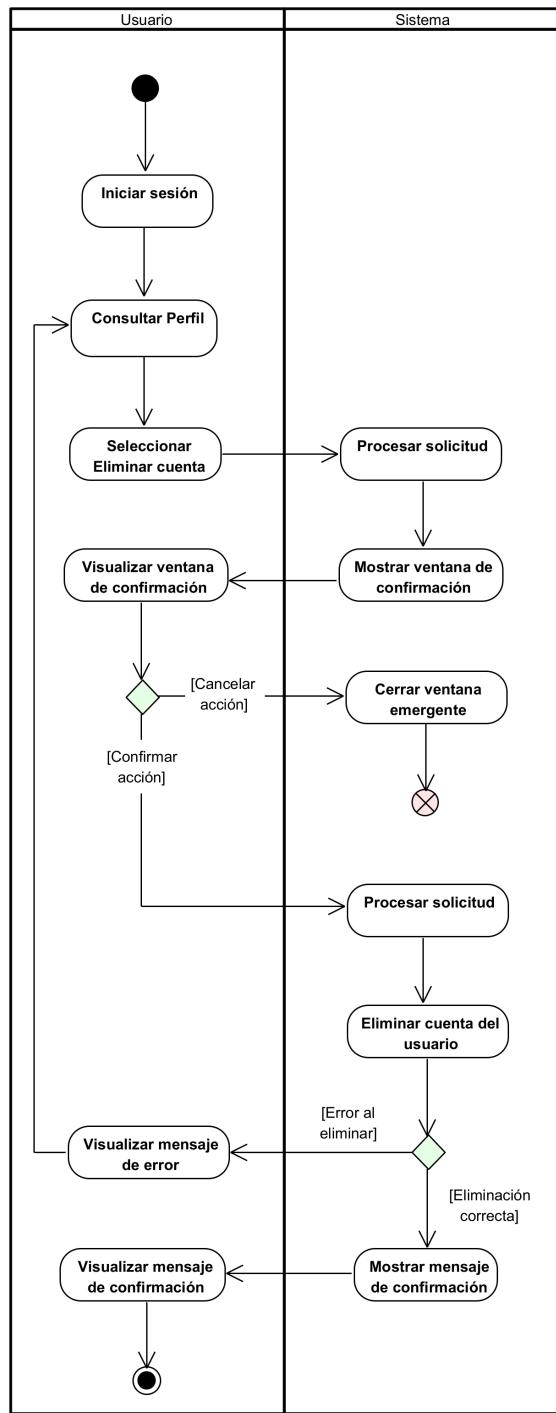
**Figura 52**

Diagrama de Actividad para Editar Perfil (RF4.2).



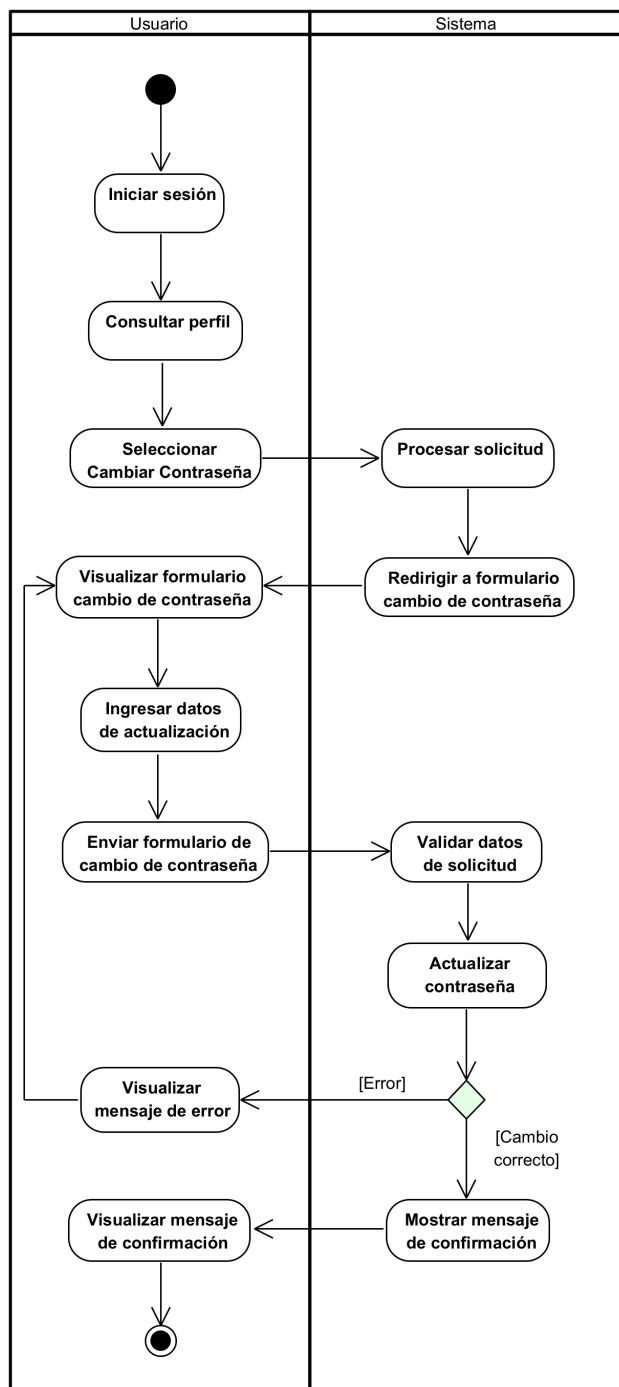
**Figura 53**

Diagrama de Actividad para Eliminar Perfil (RF4.3).



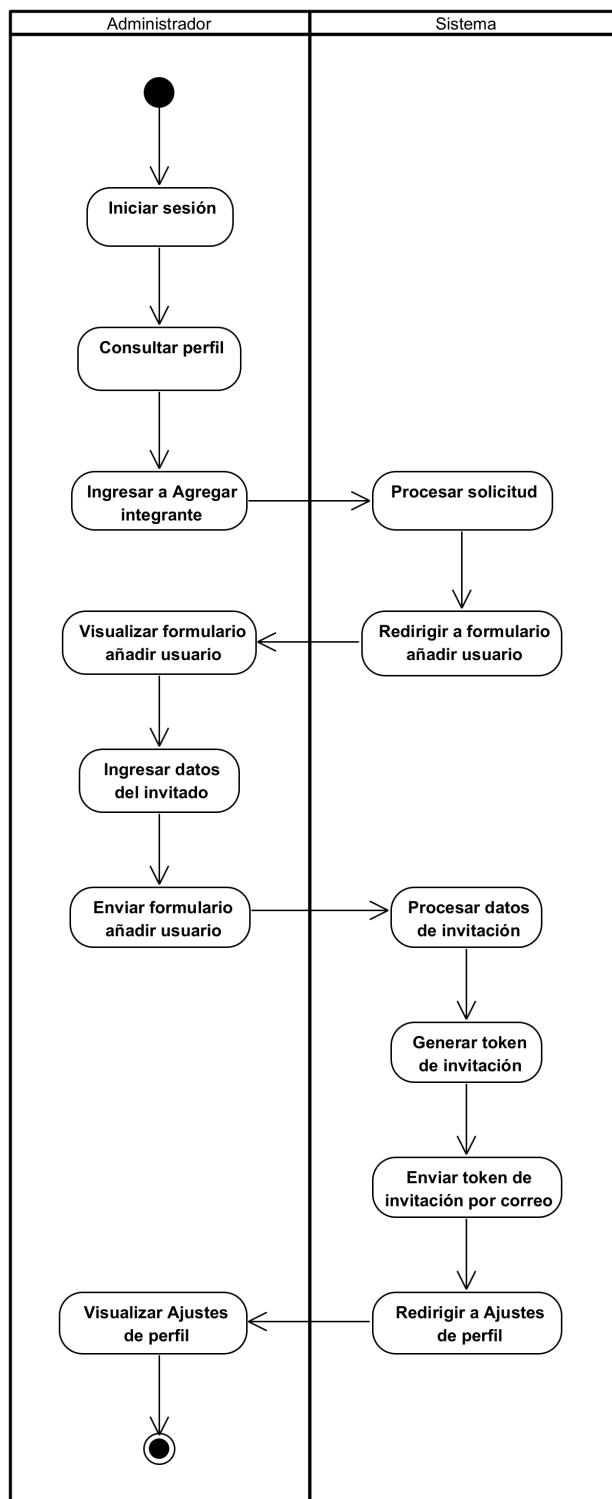
**Figura 54**

*Diagrama de Actividad para Cambiar Contraseña (RF4.4).*



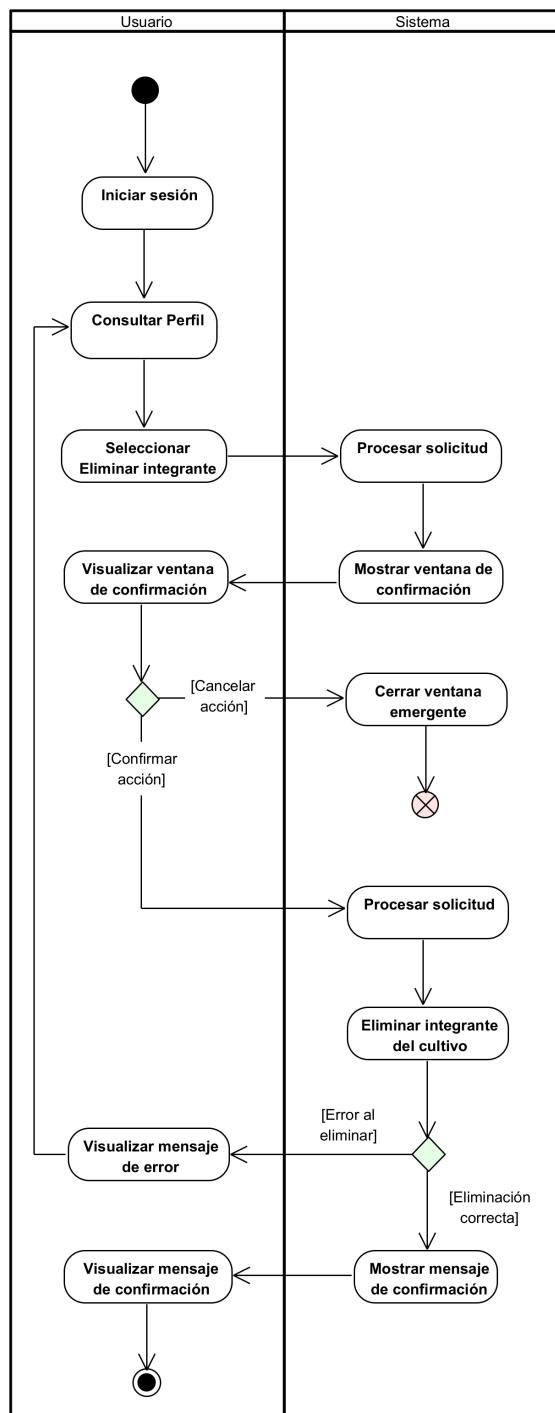
**Figura 55**

Diagrama de Actividad para Agregar Integrante de Cultivo (RF4.5).



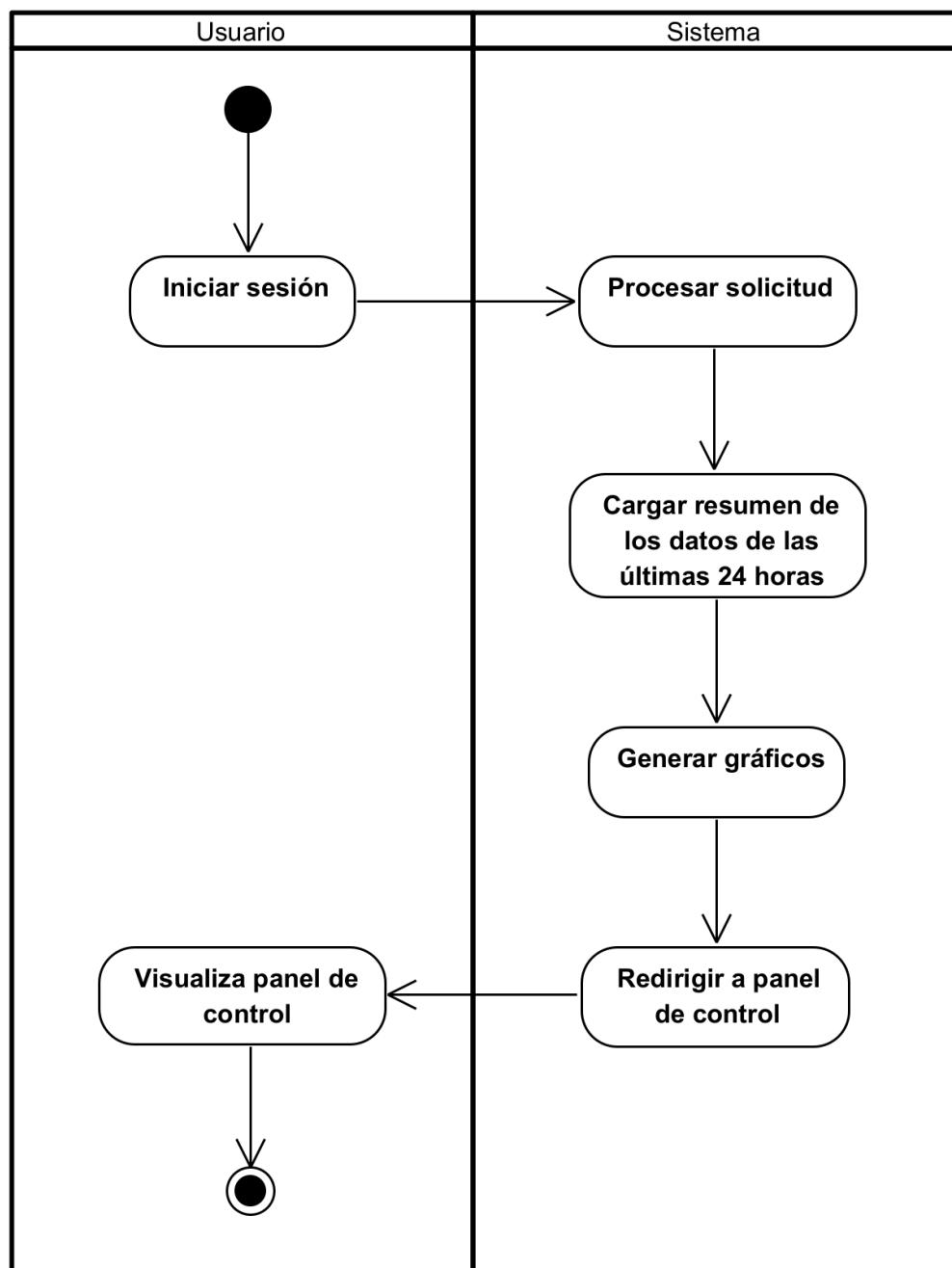
**Figura 56**

Diagrama de Actividad para Eliminar Integrante de Cultivo (RF4.6).



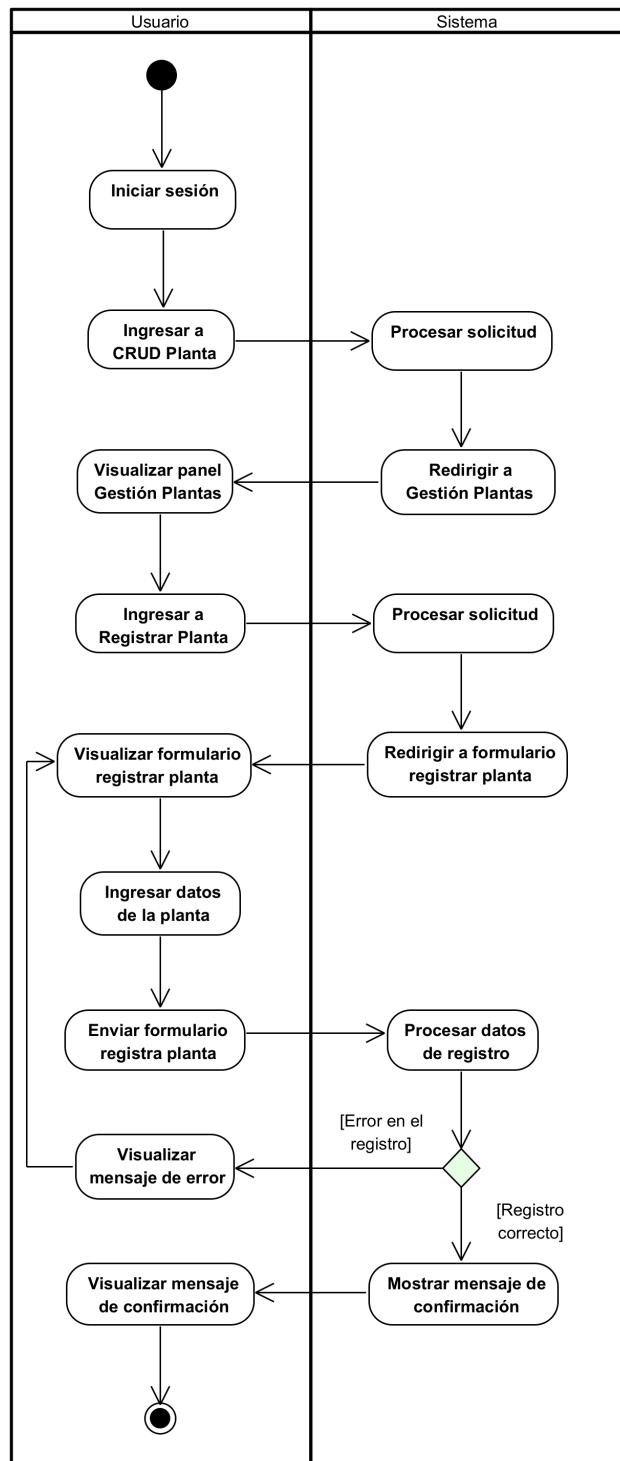
**Figura 57**

Diagrama de Actividad para el Módulo de Mediciones (RF5.0).



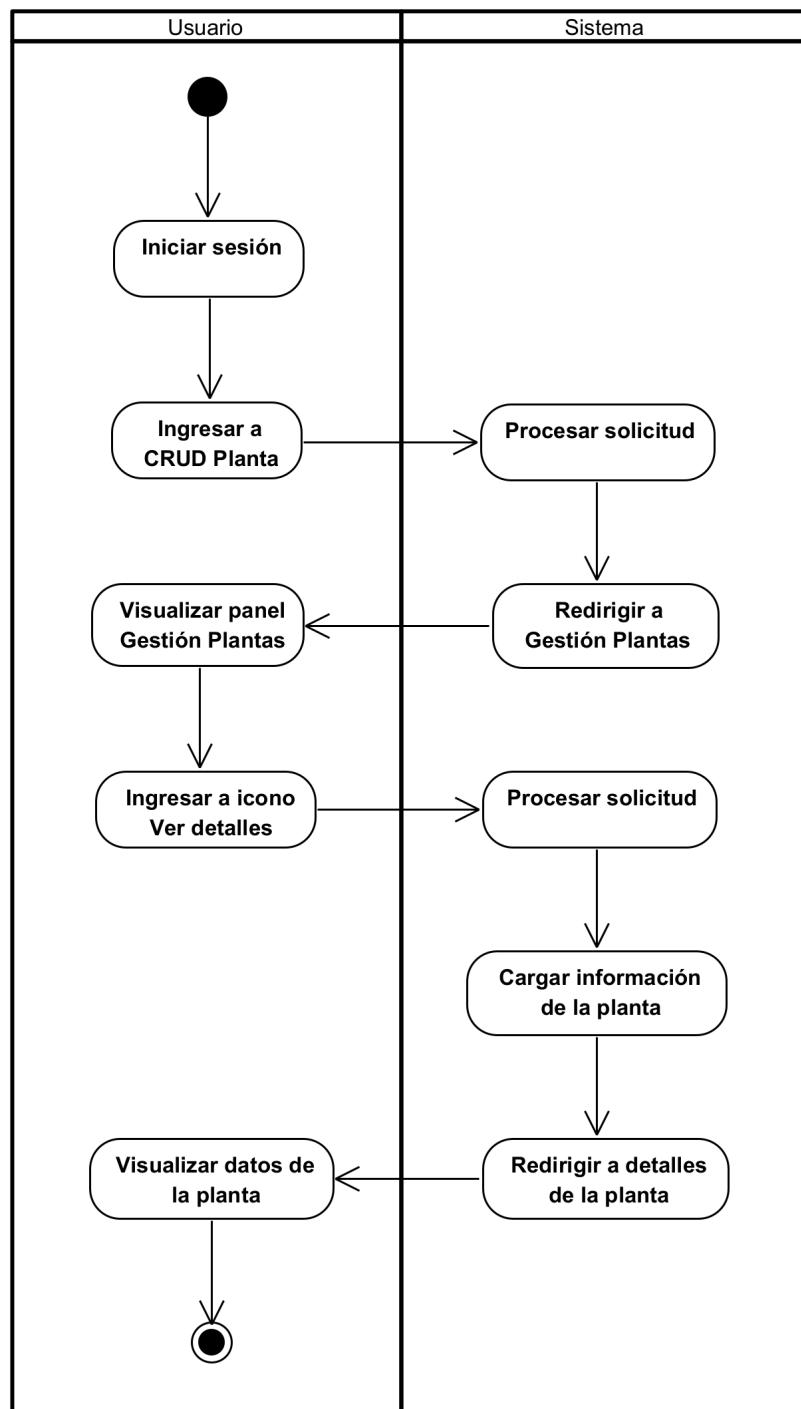
**Figura 58**

Diagrama de Actividad para Crear Planta (RF6.1).



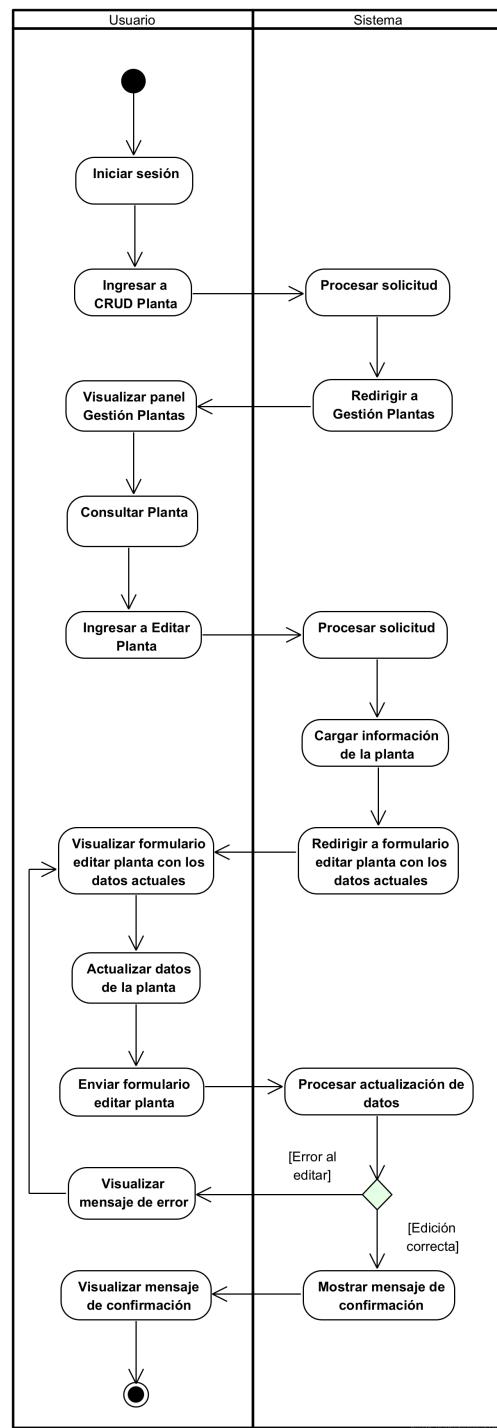
**Figura 59**

Diagrama de Actividad para Consultar Planta (RF6.2).



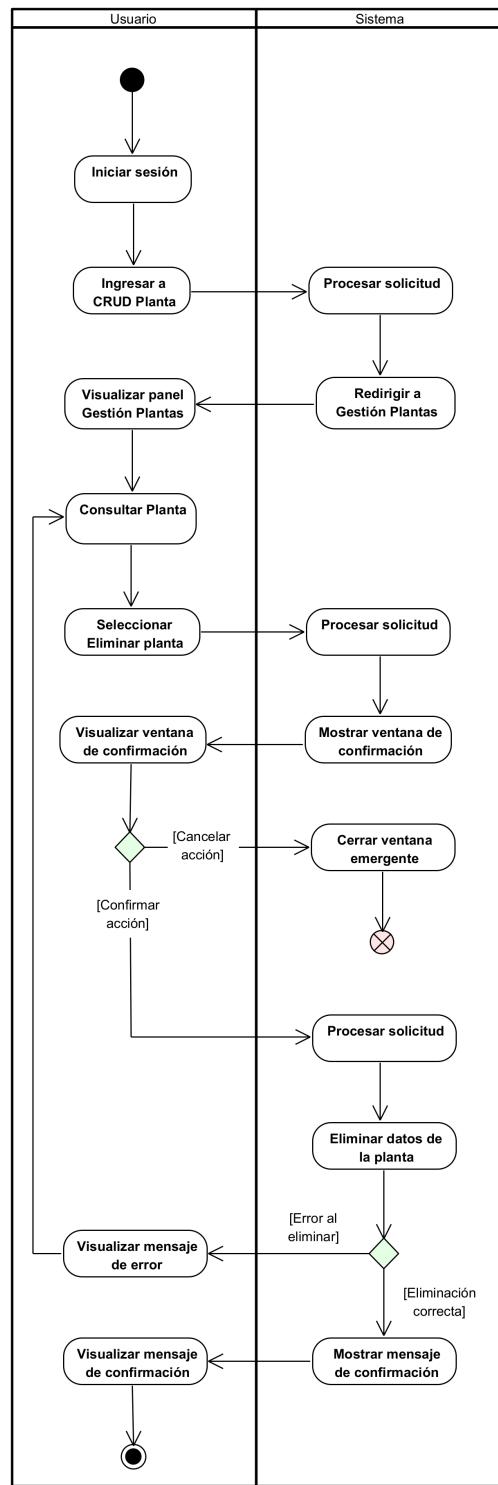
**Figura 60**

Diagrama de Actividad para Editar Planta (RF6.3).



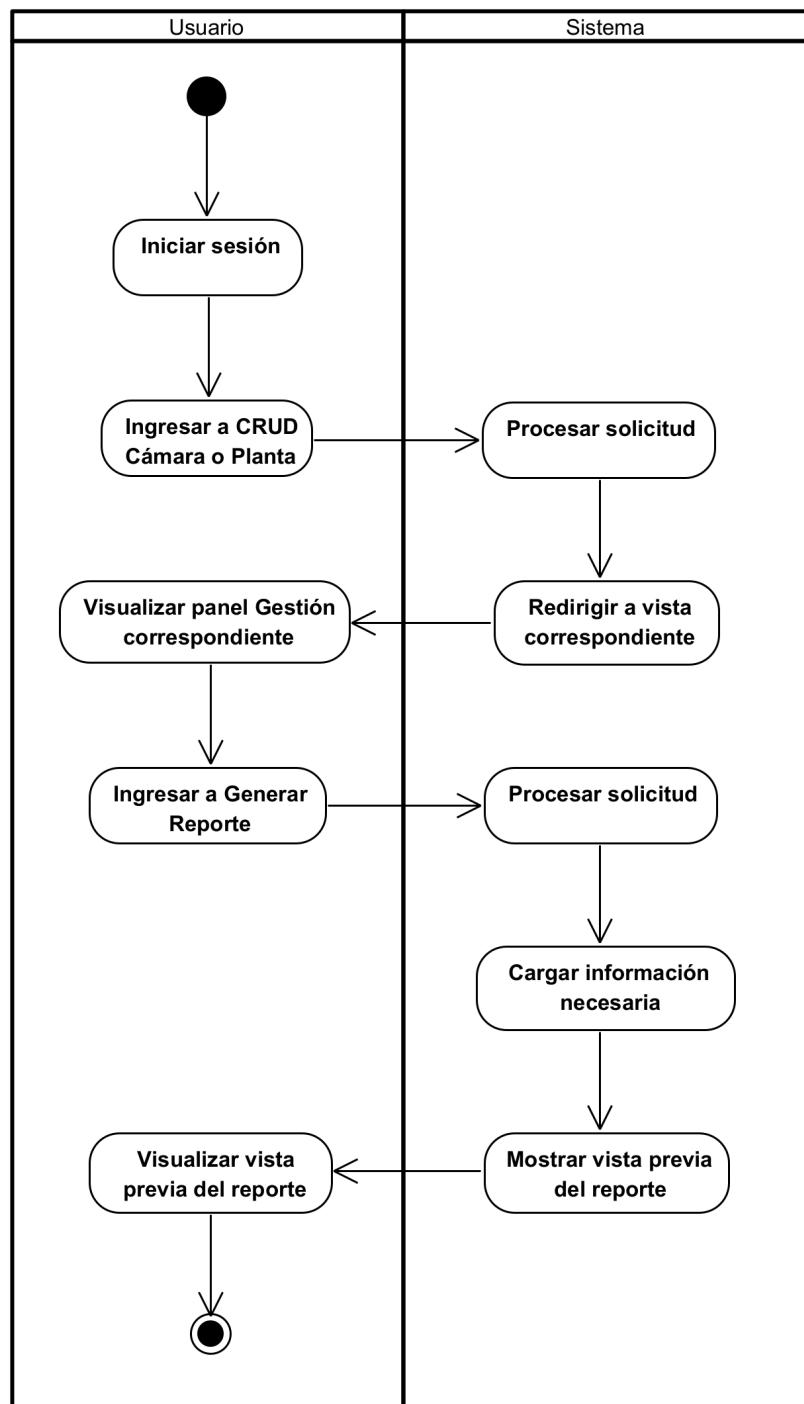
**Figura 61**

Diagrama de Actividad para Eliminar Planta (RF6.4).



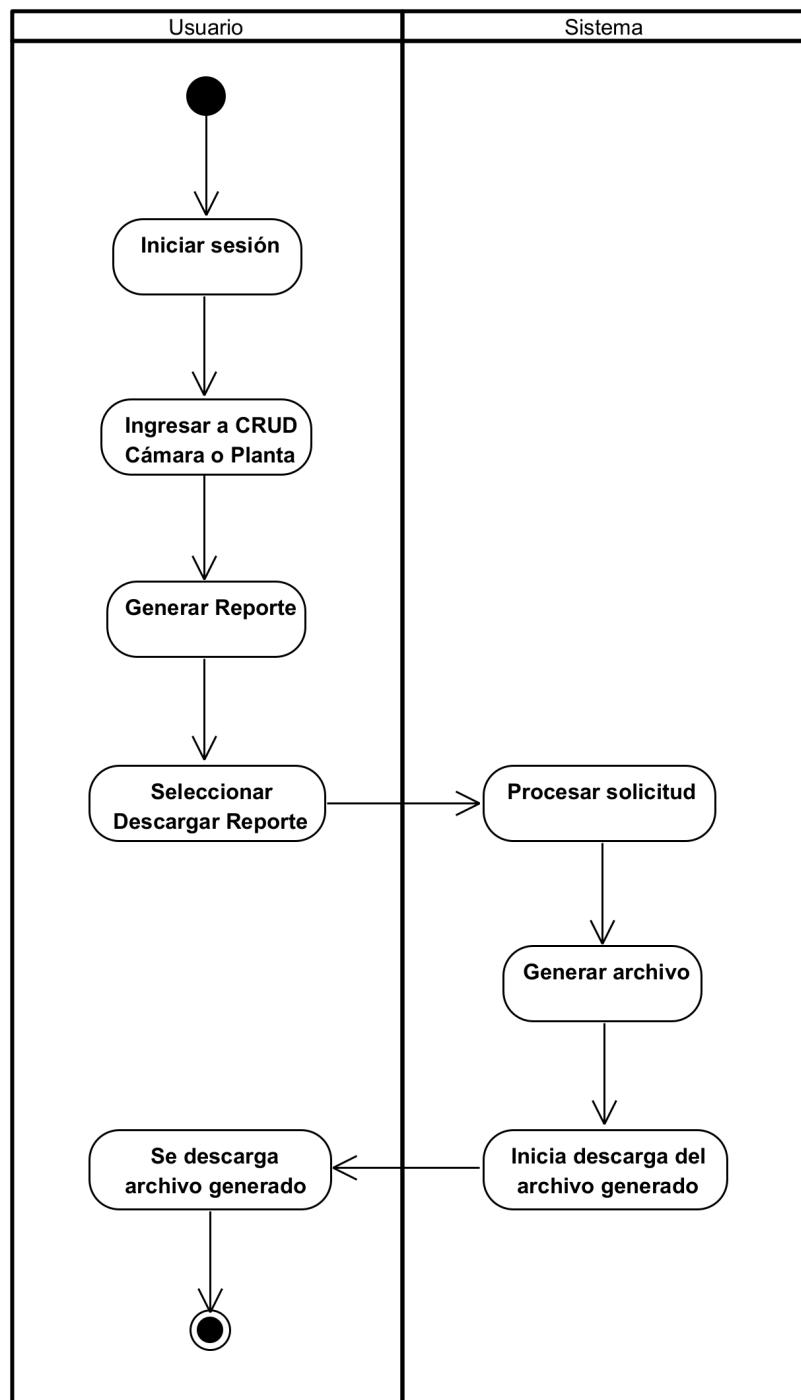
**Figura 62**

*Diagrama de Actividad para Generar Reporte (RF7.0).*



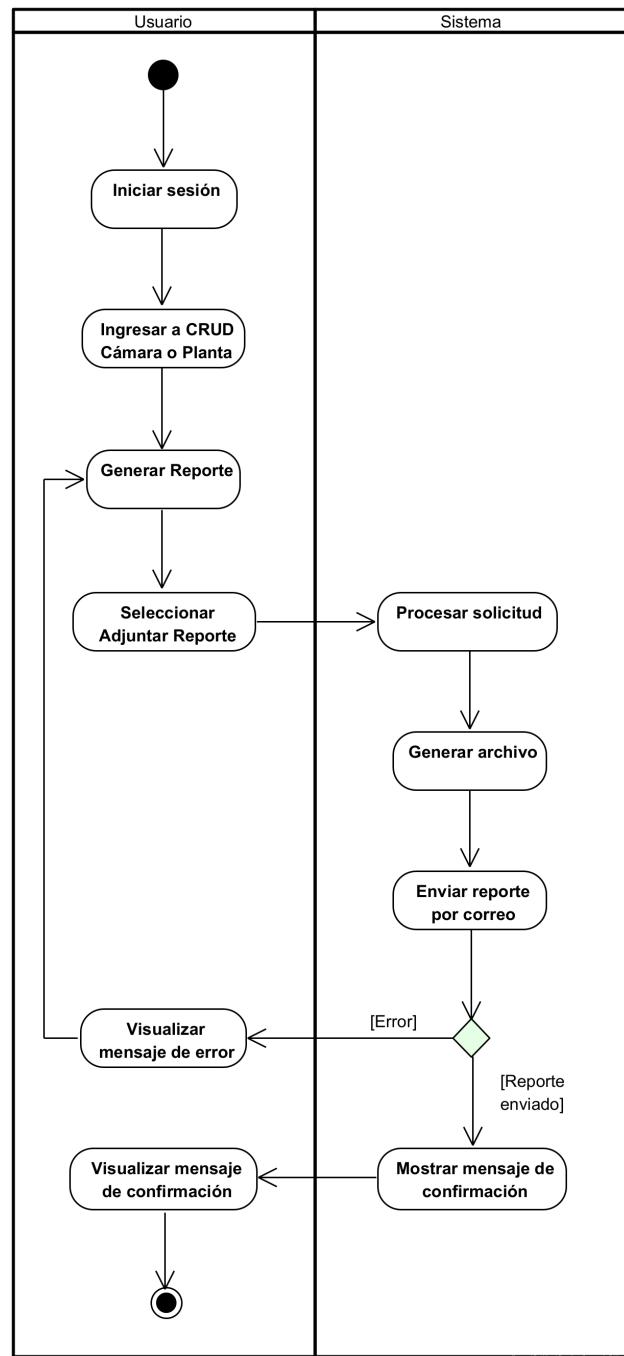
**Figura 63**

*Diagrama de Actividad para Descargar Reporte (RF7.1).*



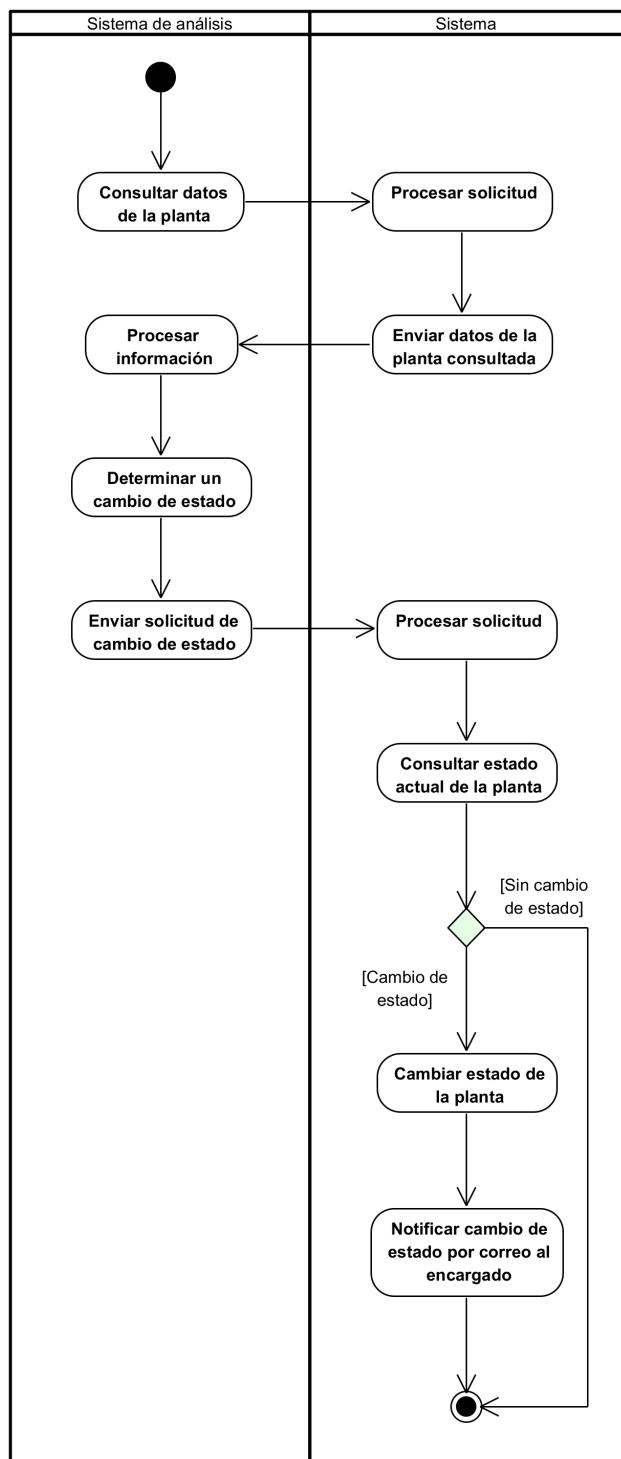
**Figura 64**

*Diagrama de Actividad para Adjuntar Reporte (RF7.2).*



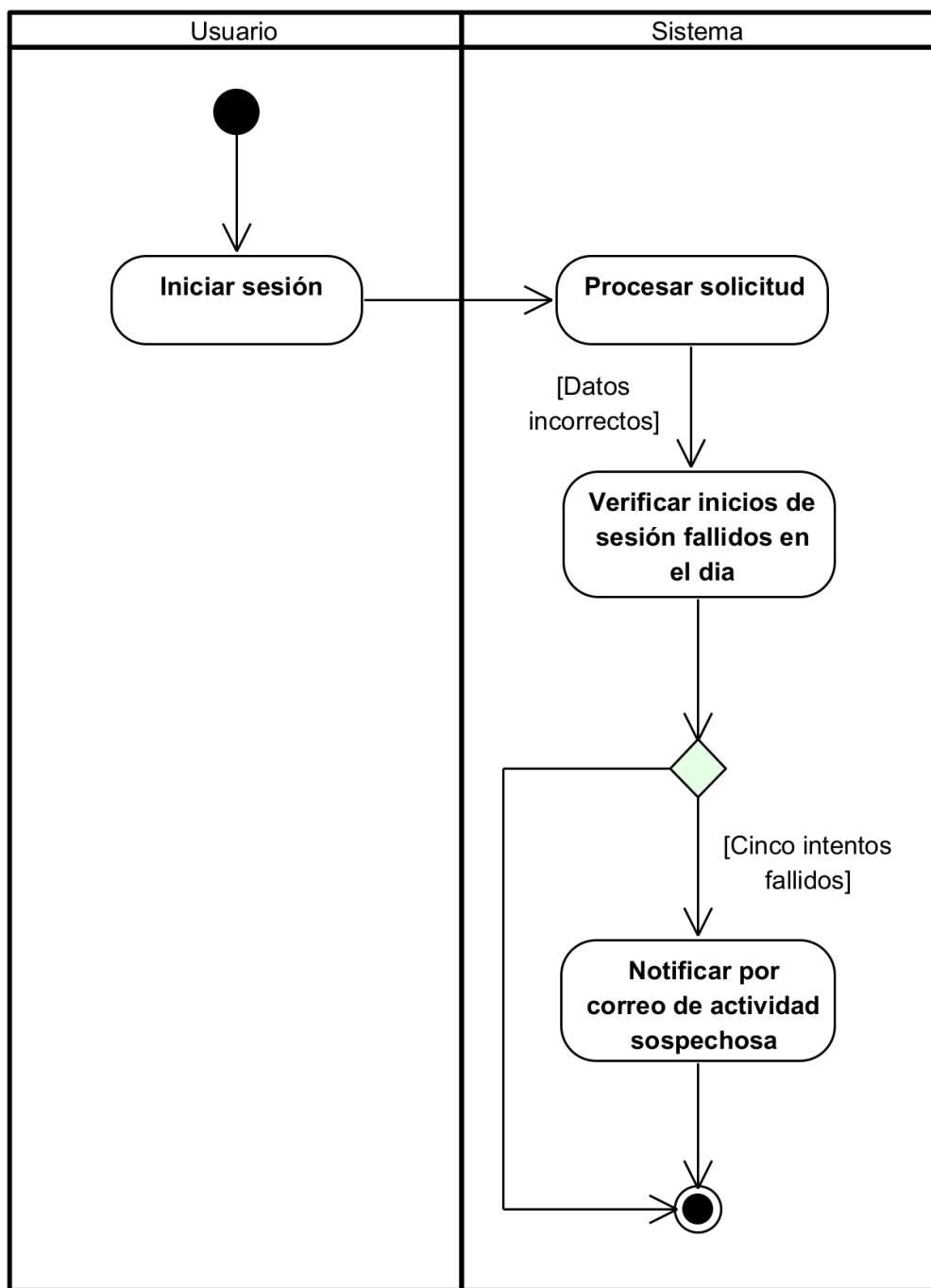
**Figura 65**

Diagrama de Actividad para Notificar Planta (RF8.1).



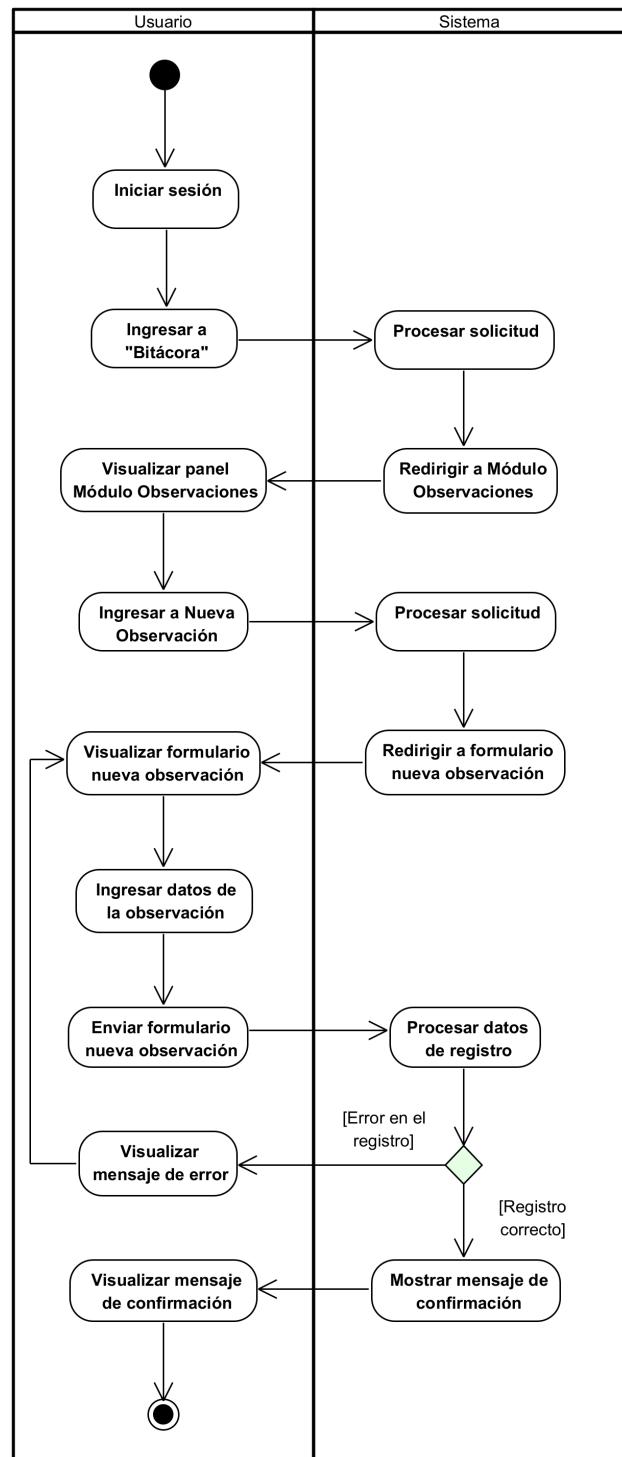
**Figura 66**

Diagrama de Actividad para Notificar Seguridad (RF8.2).



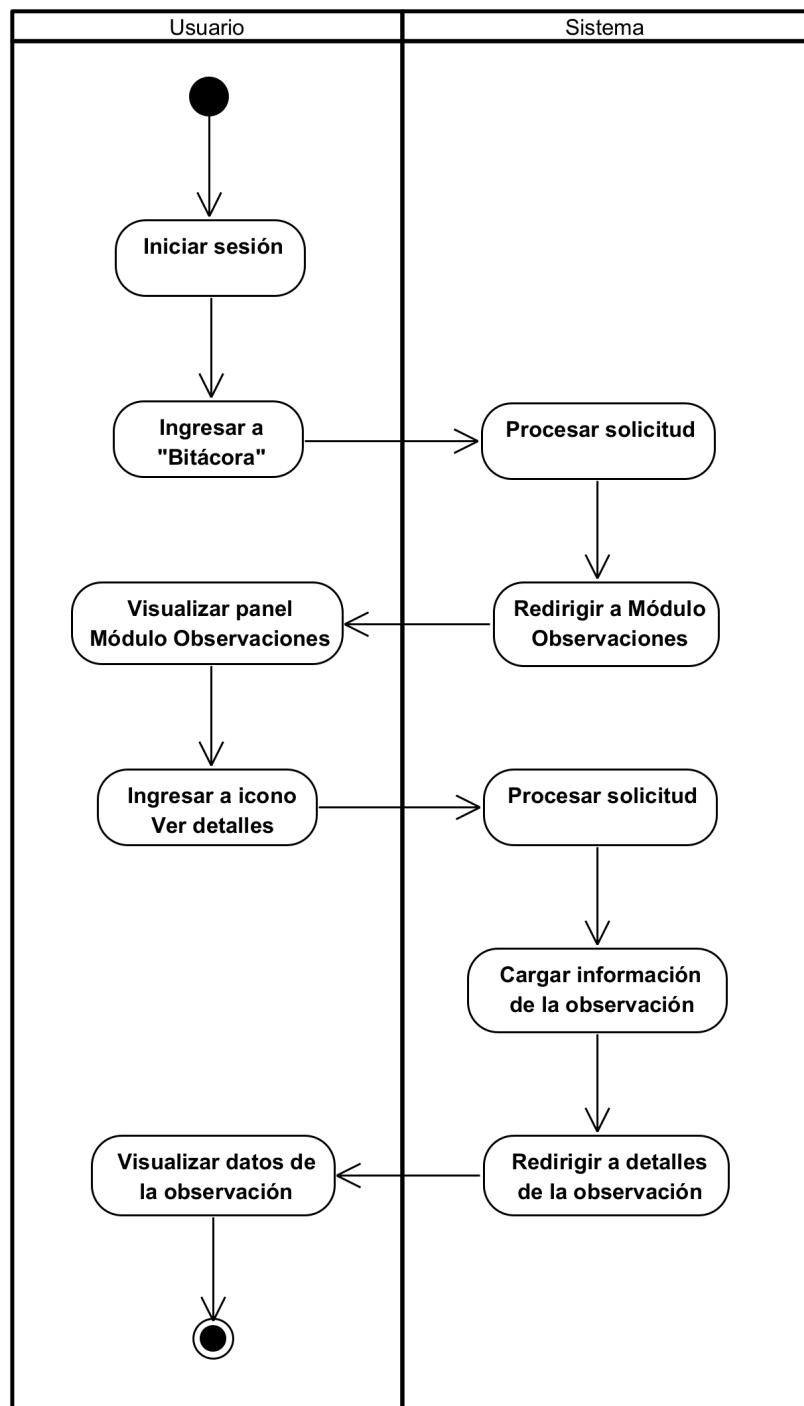
**Figura 67**

Diagrama de Actividad para Crear Observación (RF9.1).



**Figura 68**

Diagrama de Actividad para Consultar Observación (RF9.2).



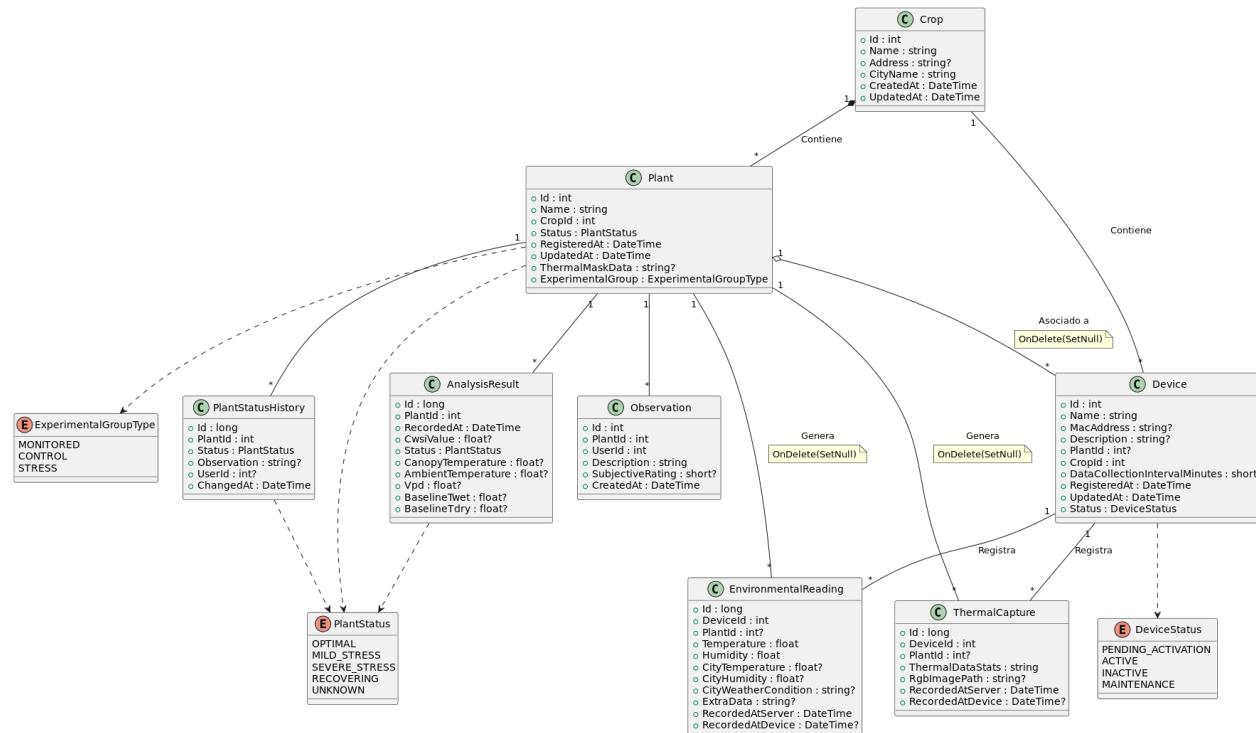
## 2.4.5. Diagrama de Clases

Una clase es la descripción de un concepto del dominio de la aplicación o del dominio de la solución (Rumbaugh y cols., 2007). Las clases son el centro alrededor del cual se organiza la vista de clases (Rumbaugh y cols., 2007). La vista estática se muestra en los diagramas de clases (Rumbaugh y cols., 2007).

En esta sección se presenta el diagrama de clases principal del sistema. Se describirán las tablas (clases) más importantes, sus atributos, las relaciones (asociaciones, generalizaciones) entre ellas y cómo estas estructuras en conjunto definen el funcionamiento general y la organización de los datos del sistema.

**Figura 69**

*Diagrama de Clases: Entidades Principales y Datos de Monitoreo.*



La Figura 69 presenta el núcleo del modelo de dominio del sistema. Se observan las entidades

centrales que representan los elementos físicos monitoreados:

- **Crop**: Representa el cultivo y actúa como la entidad contenedora principal (tenant).
- **Plant**: Modela cada planta individual dentro de un cultivo. Es una entidad central que agrega gran parte de los datos de monitoreo.
- **Device**: Representa el dispositivo físico de hardware encargado de la recolección de datos.

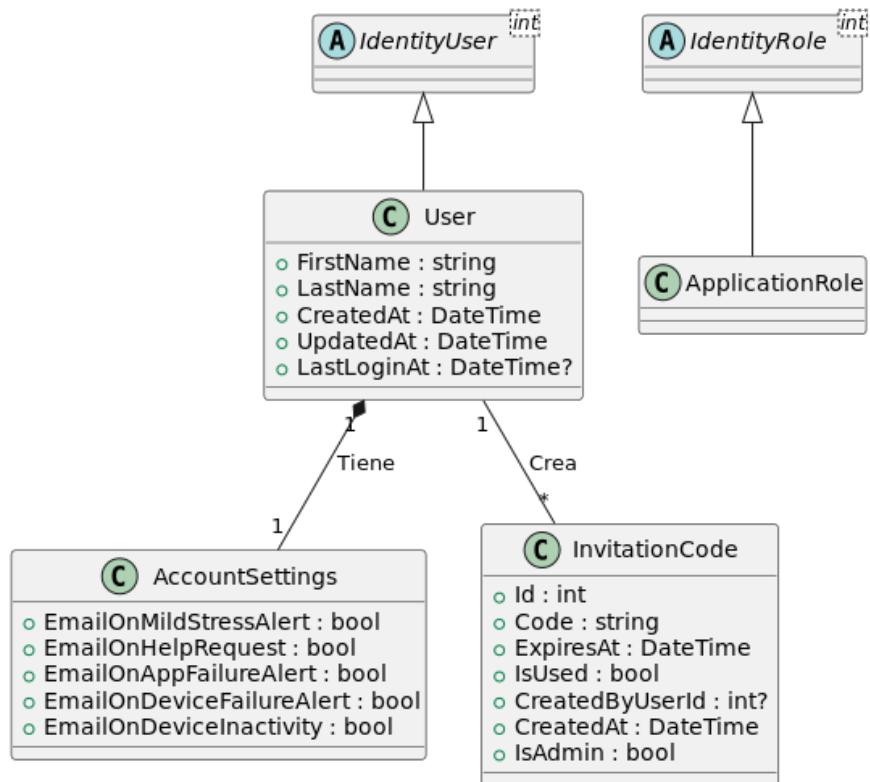
El diagrama ilustra las relaciones de composición y asociación entre estas entidades y los datos que generan o a los que están asociados:

- Un **Crop** contiene múltiples **Plant** y **Device**.
- Un **Device** está asociado a un **Crop** y opcionalmente a una **Plant** específica. La relación con **Plant** está configurada con `OnDelete(SetNull)`, lo que significa que si una planta se elimina, el campo **PlantId** en los dispositivos asociados se establecerá en nulo, pero los dispositivos no se eliminarán.
- Tanto **Plant** como **Device** están asociados a las entidades que almacenan los datos recolectados: **EnvironmentalReading** (datos de sensores ambientales) y **ThermalCapture** (datos de capturas térmicas). Las relaciones de **Plant** con estos datos también usan `OnDelete(SetNull)`.
- La entidad **Plant** también se relaciona con **AnalysisResult** (resultados del cálculo de CWSI), **PlantStatusHistory** (historial de cambios de estado) y **Observation** (anotaciones manuales de los usuarios).

Finalmente, el diagrama muestra el uso de enumeraciones (`enum`) como **PlantStatus**, **DeviceStatus** y **ExperimentalGroupType** para representar estados y clasificaciones de manera controlada y legible dentro del dominio. Se detallan todos los atributos de cada clase, especificando su tipo de dato (ej. `int`, `string`, `DateTime`, `float?`).

**Figura 70**

*Diagrama de Clases: Gestión de Usuarios y Roles.*



Continuando con el modelo de dominio, la Figura 70 ilustra las clases relacionadas con la gestión de usuarios y el control de acceso.

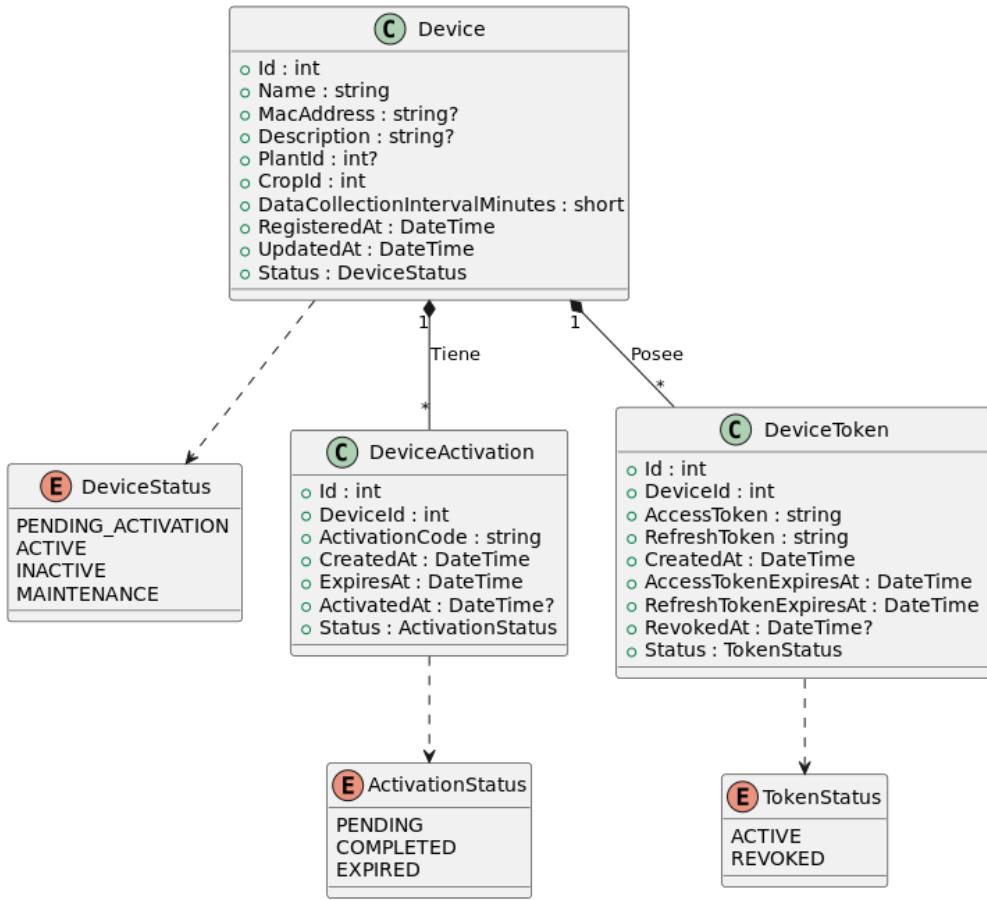
- **User**: Representa a un usuario del sistema web. Esta clase hereda de la clase base **IdentityUser<int>** proporcionada por ASP.NET Core Identity, extendiéndola con atributos personalizados como **FirstName**, **LastName**, fechas de creación/actualización y el último inicio de sesión (**LastLoginAt**).
- **ApplicationRole**: Representa un rol dentro del sistema (ej. Administrador). Hereda de **IdentityRole<int>** y, aunque no añade atributos propios, permite la integración con el sistema de roles de Identity.

- **AccountSettings**: Modela las preferencias de notificación por correo electrónico de un usuario. Existe una relación de composición fuerte (indicada por el rombo negro) entre **User** y **AccountSettings**, significando que cada usuario tiene exactamente una instancia de configuración de cuenta asociada.
- **InvitationCode**: Almacena los códigos de invitación de un solo uso para el registro de nuevos usuarios. Un **User** (administrador) puede crear múltiples códigos de invitación, representado por la asociación opcional (la relación puede tener **CreatedByUserId** nulo).

Este diagrama muestra cómo el sistema extiende las funcionalidades base de ASP.NET Core Identity para adaptarlas a las necesidades específicas del proyecto, como las configuraciones de cuenta y el flujo de invitaciones. Se detallan todos los atributos y sus tipos de dato correspondientes.

**Figura 71**

*Diagrama de Clases: Ciclo de Vida y Seguridad del Dispositivo.*



La Figura 71 detalla las clases que gestionan el ciclo de vida y la seguridad de la entidad **Device**. Este subsistema es crucial para garantizar que solo el hardware autorizado pueda interactuar con la API del sistema.

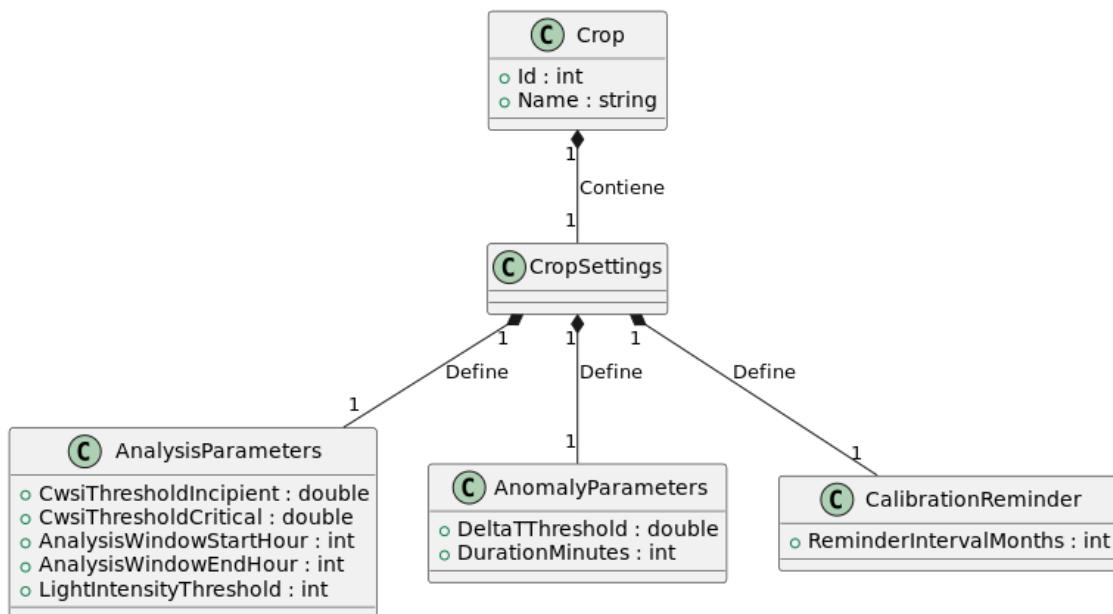
- **Device**: Es la clase central, ya presentada, que representa el dispositivo físico.
- **DeviceActivation**: Modela el proceso de activación inicial y de un solo uso de un dispositivo. Almacena el **ActivationCode**, su fecha de expiración y el estado del proceso.
- **DeviceToken**: Representa el par de tokens de autenticación (**AccessToken** y **RefreshToken**)

que se generan para un dispositivo una vez que ha sido activado. Esta clase gestiona la validez y el estado de revocación de los tokens para la autenticación continua.

El diagrama muestra una relación de composición (indicada por el rombo negro) desde **Device** hacia **DeviceActivation** y **DeviceToken**. Esto significa que la existencia de los códigos de activación y los tokens está intrínsecamente ligada a la del dispositivo; son parte fundamental de su ciclo de vida. Adicionalmente, se utilizan las enumeraciones **DeviceStatus**, **ActivationStatus** y **TokenStatus** para gestionar los estados de cada entidad de forma controlada y explícita.

**Figura 72**

*Diagrama de Clases: Configuraciones de Cultivo.*



Finalmente, la Figura 72 detalla la estructura utilizada para gestionar las configuraciones específicas de cada cultivo.

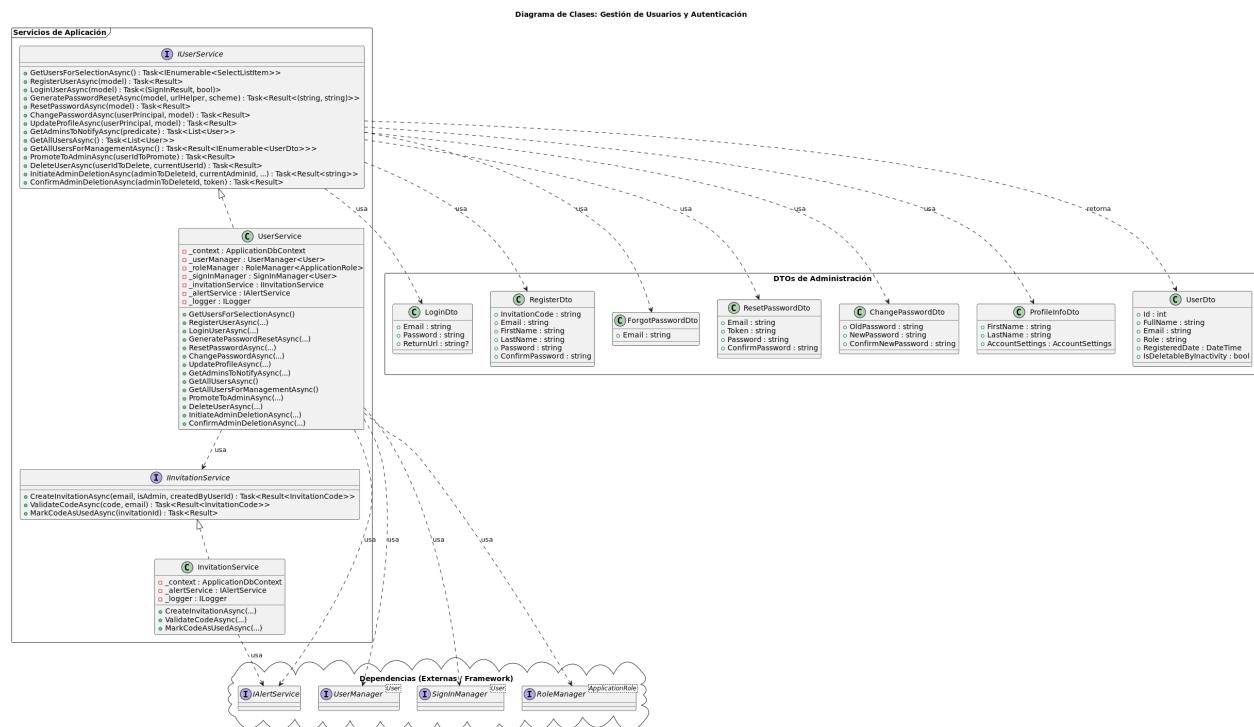
- **CropSettings**: Esta clase actúa como un contenedor principal que agrupa diferentes conjuntos de parámetros. Es parte de la entidad **Crop** a través de una relación de composición, y sus datos se almacenan en un campo de tipo **jsonb** en la base de datos.

- **AnalysisParameters**: Contiene todos los umbrales y ventanas de tiempo que controlan cómo se ejecuta el análisis de estrés hídrico (CWSI).
- **AnomalyParameters**: Define los parámetros utilizados para la detección de anomalías nocturnas en los datos de temperatura.
- **CalibrationReminder**: Especifica la frecuencia con la que se deben generar recordatorios para la calibración de los dispositivos.

La relación entre **CropSettings** y las clases de parámetros específicos (**AnalysisParameters**, etc.) es también de composición (rombo negro), indicando que estos parámetros son parte integral de la configuración general del cultivo. Esta estructura permite organizar y extender fácilmente las configuraciones del sistema en el futuro.

**Figura 73**

*Diagrama de Clases: Gestión de Usuarios y Autenticación (Capa de Aplicación).*



La Figura 73 representa el subsistema de la capa de aplicación encargado de la gestión de usuarios y la autenticación. Este diagrama ilustra las interacciones clave y las dependencias de los servicios que implementan esta lógica.

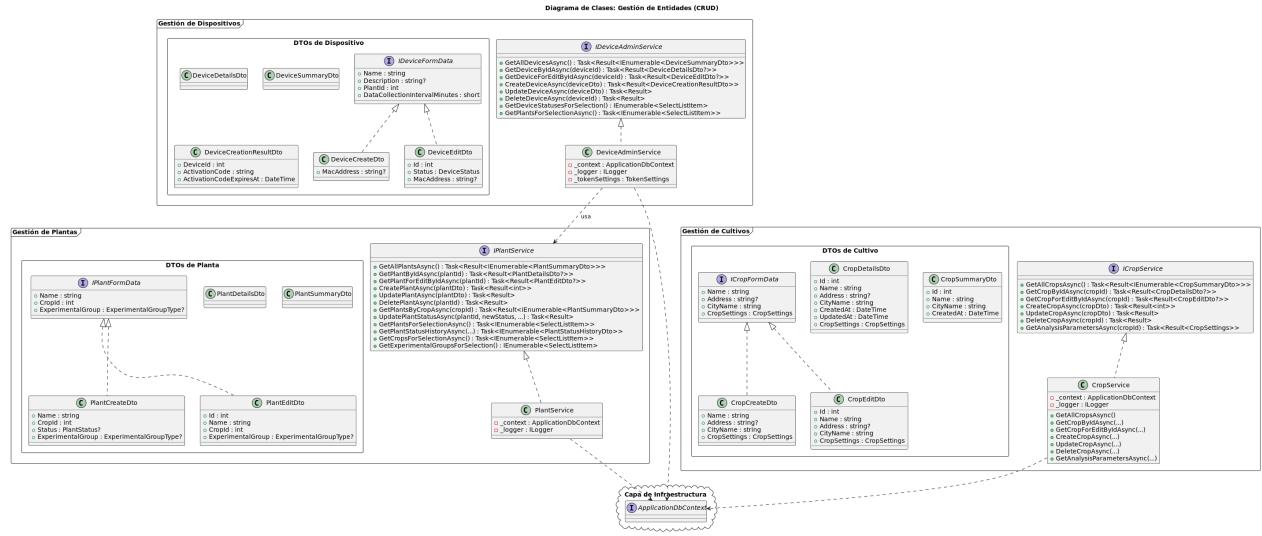
- **Servicios Principales:** Se destacan las interfaces `IUserService` e `IInvitationService`, que definen los contratos para la gestión de usuarios e invitaciones, respectivamente. Sus implementaciones concretas, `UserService` e `InvitationService`, contienen la lógica de negocio detallada.
- **Data Transfer Objects (DTOs):** Se muestra el paquete `DTOs` de `Administración`, que agrupa las clases utilizadas para transferir datos entre la capa de presentación y los servicios. Clases como `LoginDto`, `RegisterDto`, `ChangePasswordDto`, y `ProfileInfoDto` encapsulan la información necesaria para operaciones específicas como el inicio de sesión, registro, cambio de contraseña y actualización de perfil. `UserDto` se utiliza para presentar información resumida de los usuarios en las vistas de administración.
- **Dependencias:** El diagrama evidencia las dependencias de los servicios de aplicación con componentes del framework ASP.NET Core Identity (`UserManager`, `SignInManager`, `RoleManager`) para la gestión subyacente de usuarios y sesiones. También se muestra la dependencia con la interfaz `IAlertService` (definida en la capa de aplicación pero implementada en infraestructura) para el envío de notificaciones por correo electrónico.
- **Flujo de Responsabilidades:** El `UserService` actúa como orquestador principal para la mayoría de las operaciones del ciclo de vida del usuario, interactuando con los gestores de Identity y delegando la lógica de invitaciones al `IInvitationService`.

Este diagrama detalla todos los atributos y métodos públicos de las clases de servicio e interfaces, así como los atributos de los DTOs, proporcionando una vista completa de la estructura y las

responsabilidades de este módulo fundamental.

**Figura 74**

*Diagrama de Clases: Gestión de Entidades (CRUD) (Capa de Aplicación).*



La Figura 74 detalla los subsistemas dentro de la capa de aplicación responsables de las operaciones de Creación, Lectura, Actualización y Eliminación (CRUD) para las entidades principales: Cultivo, Planta y Dispositivo. El diagrama se organiza en tres paquetes lógicos principales:

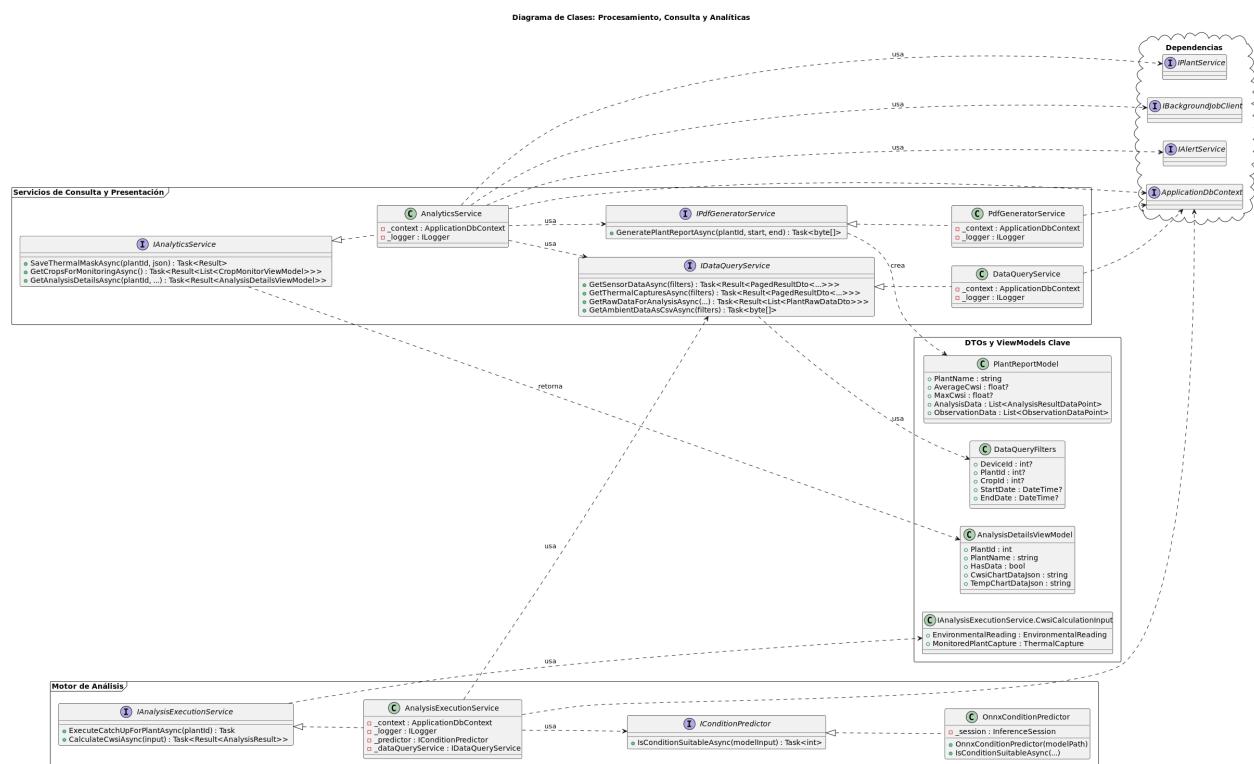
- **Gestión de Cultivos:** Incluye la interfaz **ICropService** y su implementación **CropService**, junto con los DTOs asociados (**CropCreateDto**, **CropEditDto**, **CropDetailsDto**, **CropSummaryDto**) que definen los datos necesarios para interactuar con las operaciones CRUD de los cultivos desde la capa de presentación.
- **Gestión de Plantas:** Contiene la interfaz **IPlantService** y su implementación **PlantService**, acompañados de sus DTOs correspondientes (**PlantCreateDto**, **PlantEditDto**, **PlantDetailsDto**, **PlantSummaryDto**). Este servicio gestiona el ciclo de vida de las plantas.
- **Gestión de Dispositivos (Administración):** Presenta la interfaz **IDeviceAdminService** y su implementación **DeviceAdminService**, junto con los DTOs específicos (**DeviceCreateDto**,

`DeviceEditDto, DeviceDetailsDto, DeviceSummaryDto, DeviceCreationResultDto)` para la administración de los dispositivos de hardware desde la interfaz web.

Se observa que todos estos servicios dependen principalmente del `ApplicationDbContext` (ubicado en la capa de infraestructura) para interactuar con la base de datos. Además, existen dependencias entre servicios, como la de `DeviceAdminService` hacia `IPlantService`, necesaria para poblar listas desplegables en los formularios de dispositivos. Las interfaces `ICropFormData`, `IPlantFormData`, y `IDeviceFormData` actúan como contratos comunes para los DTOs de creación y edición, promoviendo la consistencia. El diagrama incluye los atributos y métodos públicos de todas las clases e interfaces representadas.

**Figura 75**

*Diagrama de Clases: Interacción con Dispositivos y Envío de Datos (Capa de Aplicación).*



La Figura 75 ilustra la arquitectura de la API de la capa de aplicación, que sirve como punto de

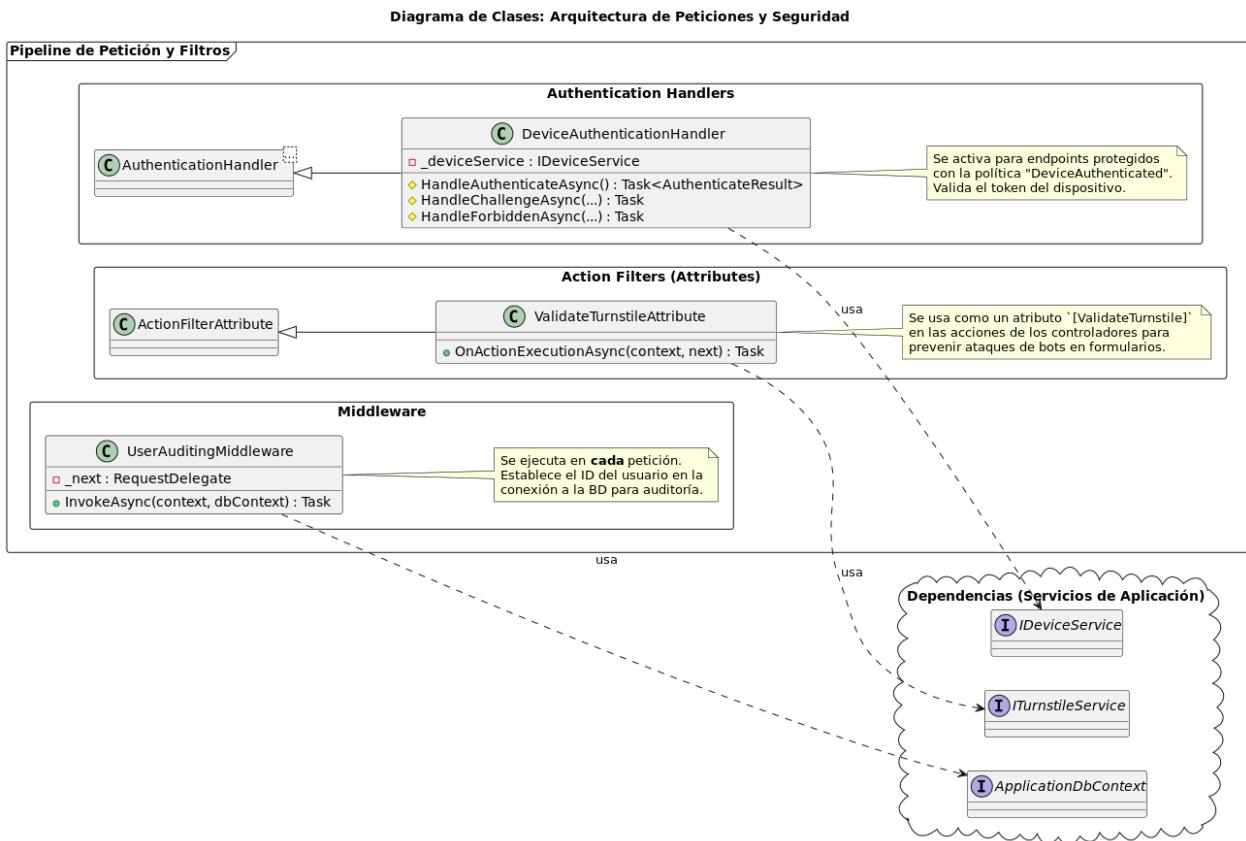
entrada para toda la comunicación proveniente de los dispositivos de hardware.

- **Servicios de API:** El diagrama muestra las dos interfaces principales que definen esta frontera de comunicación:
  - **IDeviceService:** Es responsable de todo el ciclo de vida de la autenticación del dispositivo, incluyendo la activación inicial, la validación de tokens y la generación de nuevos tokens de acceso y refresco.
  - **IDataSubmissionService:** Se encarga de recibir y procesar todos los datos enviados por los dispositivos una vez que están autenticados, como las lecturas ambientales y las capturas térmicas.
- **Data Transfer Objects (DTOs) de la API:** Se detalla un conjunto completo de DTOs que actúan como contratos de datos para la API. Clases como `DeviceActivationRequestDto`, `DeviceAuthRequestDto`, `AmbientDataDto` y `ThermalDataDto` definen la estructura exacta de los payloads (generalmente en formato JSON) que los dispositivos deben enviar en sus peticiones. Del mismo modo, DTOs como `DeviceActivationResponseDto` definen la estructura de las respuestas del servidor.
- **Dependencias de Infraestructura:** Se visualizan las dependencias de estos servicios con componentes de la capa de infraestructura. El `DataSubmissionService` depende de `IWeatherService` para enriquecer los datos y de `IFileStorageService` para almacenar imágenes. Por su parte, el `DeviceService` depende del `ApplicationDbContext` para la persistencia y de `TokenSettings` para la configuración de la duración de los tokens.

Este diagrama es clave para entender la interfaz programática que el firmware de los dispositivos de hardware debe consumir para interactuar correctamente con el sistema.

**Figura 76**

Diagrama de Clases: Arquitectura de Peticiones y Seguridad.



La Figura 76 ilustra los componentes clave de las capas de Infraestructura y Presentación que intervienen en el procesamiento y la seguridad de las peticiones HTTP entrantes, antes de que lleguen a la lógica de negocio de los controladores.

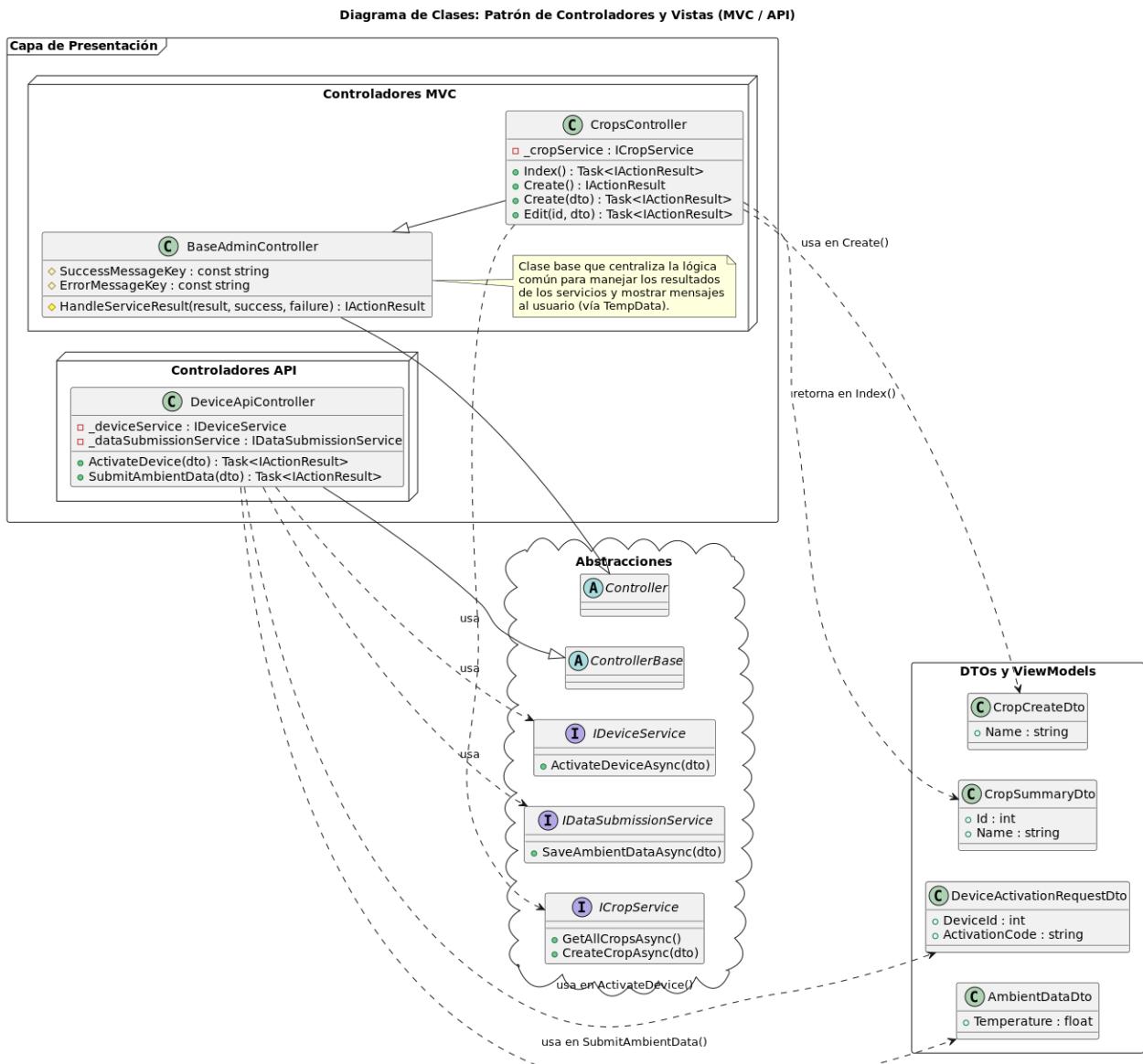
- **UserAuditingMiddleware:** Es un componente de middleware que se ejecuta en cada petición. Su función principal es extraer el identificador del usuario autenticado (si existe) y establecerlo como el `application_name` en la conexión de la base de datos (`ApplicationContext`). Esto permite una auditoría detallada a nivel de base de datos para rastrear qué usuario realizó cada operación.

- **DeviceAuthenticationHandler**: Es un manejador de autenticación personalizado que hereda de **AuthenticationHandler**. Se activa específicamente para los endpoints de la API de dispositivos protegidos con la política "DeviceAuthenticated". Utiliza el **IDeviceService** para validar el token de acceso (**AccessToken**) enviado en el encabezado de autorización y, si es válido, construye la identidad (**ClaimsPrincipal**) del dispositivo.
- **ValidateTurnstileAttribute**: Es un filtro de acción (**ActionFilterAttribute**) que se aplica como un atributo (**[ValidateTurnstile]**) a las acciones de los controladores que manejan envíos de formularios públicos. Antes de que se ejecute la acción, este filtro extrae el token de Cloudflare Turnstile de la petición y utiliza el **ITurnstileService** para verificar su validez, previniendo así ataques automatizados por bots.

Este diagrama muestra cómo estas clases interceptan las solicitudes en diferentes etapas del pipeline de ASP.NET Core (Middleware, Autenticación, Filtro de Acción) para aplicar lógica transversal de seguridad y auditoría. Se visualizan también sus dependencias clave con servicios de la capa de aplicación y el contexto de la base de datos.

**Figura 77**

Diagrama de Clases: Patrón de Controladores y Vistas (MVC / API).



Finalmente, la Figura 77 describe el patrón arquitectónico seguido por los controladores en la capa de Presentación. En lugar de mostrar cada controlador individualmente, este diagrama utiliza ejemplos representativos para ilustrar la estructura general.

- **Controladores MVC:** Se muestra la clase base `BaseAdminController`, que hereda de la clase `Controller` de ASP.NET Core. Esta clase base centraliza lógica común para todos los controladores del área de administración, como el manejo estandarizado de los resultados de los servicios (`HandleServiceResult`) y la comunicación de mensajes al usuario a través de `TempData`. Un controlador típico como `CropsController` hereda de `BaseAdminController` y depende de la interfaz del servicio correspondiente (`ICropService`) para realizar las operaciones. Utiliza DTOs como `CropCreateDto` para recibir datos de los formularios y `CropSummaryDto` (o `ViewModels`) para pasar datos a las vistas.
- **Controladores API:** Se representa con `DeviceApiController`, que hereda de  `ControllerBase`. Estos controladores están diseñados para ser consumidos por clientes programáticos (como el hardware). Dependen de las interfaces de servicio (`IDeviceService`, `IDataSubmissionService`) y utilizan DTOs específicos de la API (como `DeviceActivationRequestDto` o `AmbientDataDto`) para definir los contratos de datos de los endpoints.
- **Abstracciones y Dependencias:** El diagrama resalta cómo los controladores dependen de las interfaces de los servicios definidos en la capa de aplicación, respetando el principio de inversión de dependencias y manteniendo el acoplamiento bajo. También muestra cómo los DTOs actúan como la interfaz de datos entre la capa de presentación y la capa de aplicación.

Este patrón asegura que los controladores sean ligeros, centrados en manejar las peticiones HTTP y orquestar las llamadas a la lógica de negocio, manteniendo una clara separación de responsabilidades dentro de la arquitectura.

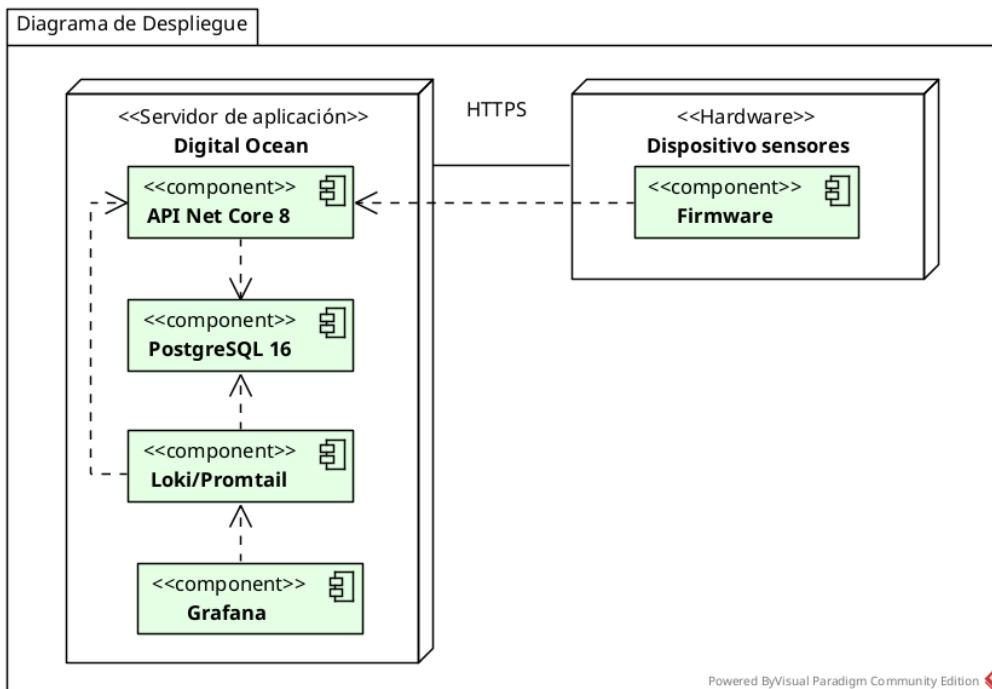
#### 2.4.6. Diagrama de Despliegue

La vista de despliegue muestra la disposición física de los nodos, que son recursos computacionales de tiempo de ejecución, como computadoras u otros dispositivos (Rumbaugh y cols., 2007).

Durante la ejecución, los nodos pueden contener artefactos, que son entidades físicas como archivos (Rumbaugh y cols., 2007).

**Figura 78**

*Diagrama de Despliegue del Sistema.*



El Diagrama de Despliegue, mostrado en la Figura 78, representa la arquitectura física del sistema y cómo sus componentes de software se distribuyen en diferentes nodos de hardware y servidores. Se identifican dos nodos principales:

- «Servidor de aplicación» Digital Ocean: Representa el nodo principal del servidor que aloja la aplicación. Este servidor contiene un conjunto de componentes de software que operan de forma integrada mediante Docker Compose:
  - API Net Core 8: Es el componente de aplicativo que gestiona la lógica de negocio y provee las vistas al usuario.

- PostgreSQL 16: El sistema gestor de base de datos donde se almacenan los datos. Contiene PgAudit para proveer la auditoria de la base de datos al sistema de Logging.
  - Loki/Promtail: Componentes destinados a la recopilación y gestión de logs.
  - Grafana: Herramienta de visualización y monitoreo que consume los datos de Loki/Promtail.
- «Hardware» Dispositivo sensores: Representa el dispositivo físico encargado de la toma de datos y ejecuta su propio componente de Firmware.

La comunicación entre el Dispositivo sensores (específicamente su Firmware) y la API Net Core 8 se establece a través del protocolo seguro HTTPS.

Internamente, dentro del servidor Digital Ocean, el diagrama ilustra las dependencias: la API interactúa con la base de datos PostgreSQL y, junto con esta, envía información (logs) al sistema Loki/Promtail, el cual a su vez alimenta a Grafana para la visualización.

## 2.5. Diseño de los Casos de Prueba

Texto sobre el diseño de casos de prueba utilizando SonarQube.

## 2.6. Estimación de Recursos

Texto sobre la estimación de recursos utilizando el método de puntos de función o puntos de casos de uso.

## 2.7. Resultados de la Implementación del Software

Texto sobre el resultado de implementar el software.

## **2.8. Conclusiones y Recomendaciones del software**

Discusión, conclusiones y recomendaciones sobre el software y su integración.

# **III. DOCUMENTACIÓN HARDWARE**

## **3.1. Introducción**

En este capítulo se presenta la importancia del hardware en el sistema de detección temprana de *Botrytis cinerea*, justificando la selección de cada componente y su contribución al funcionamiento global del sistema.

## **3.2. Objetivos**

- Describir los componentes seleccionados para el sistema de detección.
- Detallar la metodología utilizada para la caracterización y validación de cada componente.
- Presentar los resultados obtenidos en la integración del hardware y su rendimiento en condiciones experimentales.

## **3.3. Descripción de Componentes**

### **3.3.1. Microcontrolador ESP32-S3-WROOM-1 N16R8**

Descripción del microcontrolador y sus características principales.

### **3.3.2. Cámara termográfica MLX90640**

Descripción del sensor de termografía, sus especificaciones y función en la detección.

### **3.3.3. Sensor de luz BH1750**

Detalle del sensor de luminosidad y su relevancia en el monitoreo ambiental.

### **3.3.4. Sensor de humedad y temperatura DHT22**

Explicación del sensor de temperatura y humedad, resaltando su precisión y rango de medición.

### **3.3.5. Cámara RGB OV2640**

Breve descripción de la cámara adicional y sus aplicaciones en el proyecto.

### **3.3.6. Regulador de voltaje LM2596**

Descripción del regulador de voltaje y su importancia para garantizar la estabilidad del sistema.

## **3.4. Metodología de Caracterización**

Esta sección describe el proceso sistemático para evaluar y validar el desempeño de cada componente y su integración en el sistema final.

### **3.4.1. Evaluación y verificación de componentes**

- **Definición de Parámetros de Evaluación:** Establecer los parámetros (por ejemplo, precisión, tiempo de respuesta y estabilidad) a medir para cada sensor y módulo.

- **Diseño de Protocolos de Prueba:** Elaborar procedimientos detallados para realizar pruebas de calibración y verificación en condiciones controladas.
- **Implementación de Ensayos Experimentales:** Ejecutar las pruebas en laboratorio, documentando condiciones ambientales, configuraciones y resultados obtenidos.
- **Análisis de Resultados:** Comparar los datos obtenidos con las especificaciones del fabricante y los requerimientos del proyecto.

### **3.4.2. Configuración e Integración del Firmware**

- **Diseño del Esquema de Integración:** Elaborar diagramas que muestren la conexión física entre el ESP32 y cada componente, indicando rutas de comunicación (I2C, SPI, etc.).
- **Configuración en Platform.io IDE:** Documentar el proceso de configuración, instalación de librerías, asignación de pines y gestión de interrupciones o tiempos de muestreo.
- **Pruebas de Integración:** Realizar pruebas para verificar la comunicación y correcta transmisión de datos entre el hardware y el firmware.
- **Documentación de la Integración:** Registrar todos los pasos y resultados obtenidos para facilitar futuras revisiones o ajustes.

### **3.4.3. Validación y Análisis de Resultados**

En esta sección se evalúa el rendimiento global del sistema integrado mediante la comparación de datos experimentales controlados con los objetivos del proyecto. Se presentan datos, gráficos y análisis que demuestran el desempeño de cada componente y del sistema en conjunto, contrastándolos con los parámetros de referencia y registrando hallazgos para futuras revisiones.

### **3.5. Implementación del Sistema Integrado**

Se describe la implementación práctica, incluyendo la integración final del hardware con el firmware, la calibración de sensores y la ejecución de pruebas en condiciones reales para ajustar y optimizar el sistema.

### **3.6. Resultados**

Interpretación de los resultados finales obtenidos, identificando fortalezas y áreas de mejora, y estableciendo la relación entre el rendimiento del hardware y los objetivos del proyecto.

# **IV. ESTUDIO EXPERIMENTAL**

## **4.1. Introducción**

La gestión eficiente del agua en la agricultura se ha convertido en un desafío crítico, intensificado por los efectos del cambio climático y la creciente demanda de recursos hídricos (Salgado Vargas, Sánchez-García, Volke-Haller, y Colinas León, 2018). Para cultivos de alto valor como el arándano (var. Biloxi), el estrés hídrico representa una de las principales amenazas para la productividad y la calidad del fruto. La detección temprana de esta condición es fundamental para optimizar las prácticas de riego y asegurar la sostenibilidad del cultivo.

La termografía infrarroja (IRT) ha demostrado ser una herramienta eficaz para el monitoreo no invasivo del estado hídrico de las plantas (Jones, 2004; Poblete-Echeverría, Fuentes, Ortega-Farias, Gonzalez-Talice, y Yuri, 2023), basándose en la premisa de que la temperatura de la superficie foliar aumenta cuando la planta cierra sus estomas para conservar agua (Rinza, Ramírez, y Ninanya, 2021). Sin embargo, el alto costo de los equipos termográficos comerciales ha limitado su adopción, creando una brecha tecnológica especialmente para pequeños y medianos productores.

Este capítulo detalla el estudio experimental realizado para validar un sistema de termografía de bajo costo como una solución viable para el monitoreo del estrés hídrico en el arándano. Se describe el diseño, la construcción y la validación metrológica de un prototipo de hardware, así como la metodología empleada para inducir, monitorear y analizar los efectos del estrés hídrico en una

planta de arándano en condiciones controladas. El objetivo final de este experimento es demostrar la sensibilidad y fiabilidad del sistema propuesto para registrar patrones térmicos coherentes con la respuesta fisiológica de la planta, sentando las bases para su aplicación a mayor escala en la agricultura de precisión.

## **4.2. Objetivos del Estudio Experimental**

### **4.2.1. Objetivo General**

Validar la viabilidad y sensibilidad de un prototipo de termografía de bajo costo para detectar y monitorear la respuesta fisiológica del arándano (var. Biloxi) sometido a un ciclo de estrés hídrico controlado.

### **4.2.2. Objetivos Específicos**

1. Caracterizar y validar metrológicamente el prototipo de cámara termográfica de bajo costo para establecer su fiabilidad como instrumento de medición.
2. Registrar de forma continua la temperatura foliar y las variables ambientales (temperatura ambiente, humedad relativa) durante un ciclo completo de estrés hídrico y posterior recuperación.
3. Calcular el Índice de Estrés Hídrico del Cultivo (CWSI) a partir de los datos recopilados para cuantificar el nivel de estrés de la planta.
4. Analizar la correlación entre los datos térmicos obtenidos por el prototipo y la respuesta fisiológica esperada de la planta para demostrar la prueba de concepto del sistema.

## 4.3. Materiales y Métodos

Para la realización de este estudio, se diseñó y construyó un sistema de adquisición de datos compuesto por un módulo de hardware central y una serie de sensores.

### 4.3.1. Arquitectura del Hardware

El prototipo se basa en un diseño modular centrado en el microcontrolador ESP32-S3, seleccionado por su capacidad de procesamiento de doble núcleo, conectividad Wi-Fi integrada y bajo consumo energético. La arquitectura del dispositivo se compone de los siguientes elementos clave:

- **Microcontrolador (MCU):** Se utilizó un modelo LOLIN S3 de WeMos, que integra el microcontrolador ESP32-S3. Este componente se encarga de gestionar los sensores, procesar los datos y transmitirlos a través de la red Wi-Fi.
- **Sensor Térmico:** El núcleo del sistema es la matriz de sensores infrarrojos **MLX90640** (Melexis, 2021). Este sensor ofrece una resolución de 32x24 píxeles, proporcionando una matriz de 768 puntos de temperatura independientes. Opera en un rango de temperatura de -40 °C a 300 °C con una precisión de ±1.5 °C, lo que es adecuado para aplicaciones agrícolas.
- **Sensores Ambientales:** Para contextualizar las mediciones térmicas, se integró un sensor **BME280** que mide la temperatura ambiente, la humedad relativa y la presión barométrica (Bosch Sensortec, 2018). Adicionalmente, un sensor **BH1750** mide la iluminancia ambiental. Estos datos son cruciales para el cálculo de índices como el CWSI.
- **Almacenamiento y Alimentación:** El sistema incluye una ranura para tarjetas microSD para el almacenamiento local de datos en caso de pérdida de conexión Wi-Fi, garantizando la

integridad de la información. La alimentación del prototipo se realiza a través de un puerto USB-C, compatible con fuentes de 5V DC.

El ensamblaje se montó en una placa de circuito impreso (PCB) diseñada a medida para asegurar la estabilidad de las conexiones y la robustez del dispositivo en condiciones de campo.

#### **Figura 79**

*Diagrama del prototipo de hardware utilizado en el experimento.*

#### **4.3.2. Validación Metrológica del Sensor Térmico**

Antes del experimento con la planta, se llevó a cabo una validación metrológica del sensor térmico MLX90640 para cuantificar su precisión, siguiendo los principios del Vocabulario Internacional de Metrología (JCGM, 2008). Se realizaron pruebas de calibración en un entorno controlado utilizando un cuerpo negro de referencia y una termocupla de alta precisión. La precisión del sensor se evaluó calculando el Error Absoluto Medio (MAE) y la Raíz del Error Cuadrático Medio (RMSE), confirmando que los errores se encontraban dentro de los límites especificados por el fabricante y eran aceptables para la aplicación (Rodríguez-Alonso y Cabrejo-Paredes, 2022).

### **4.4. Diseño Experimental**

El experimento se diseñó como una prueba de concepto para evaluar la capacidad del sistema en el monitoreo de un solo individuo de arándano (var. Biloxi).

- **Sujeto Experimental:** Se utilizó una planta de arándano de la variedad Biloxi, una de las más cultivadas en la región. La planta se mantuvo en un microinvernadero para controlar las condiciones ambientales y aislarla de la lluvia.

- **Ubicación y Duración:** El estudio se llevó a cabo durante 55 días consecutivos, entre agosto y octubre de 2024, en el municipio de Facatativá, Cundinamarca, Colombia.
- **Tratamiento de Estrés Hídrico:** El experimento se dividió en tres fases:
  1. **Fase de Aclimatación (Días 1-15):** La planta recibió riego normal y constante para asegurar que partiera de un estado hídrico óptimo y se adaptara a las condiciones del microinvernadero.
  2. **Fase de Estrés Hídrico Inducido (Días 16-45):** Se suspendió completamente el riego para inducir un estado de estrés hídrico progresivo.
  3. **Fase de Recuperación (Días 46-55):** Se reanudó el riego de manera normal para observar la capacidad de recuperación de la planta y la respuesta del sistema de monitoreo.

El prototipo de hardware se instaló a una distancia fija de la planta, asegurando que el dosel foliar estuviera siempre dentro del campo de visión del sensor térmico.

## 4.5. Recolección y Análisis de Datos

La adquisición de datos se realizó de forma automatizada y continua durante los 55 días del experimento.

### 4.5.1. Recolección de Datos

El firmware del ESP32-S3 fue programado para capturar datos de todos los sensores a intervalos regulares de 15 minutos. En cada intervalo, se registraron los siguientes datos:

- Una matriz de 768 valores de temperatura del sensor térmico MLX90640.
- Temperatura ambiente ( $^{\circ}\text{C}$ ) del sensor BME280.

- Humedad relativa (%) del sensor BME280.
- Iluminancia (lux) del sensor BH1750.

Los datos fueron transmitidos en formato JSON a un servidor para su almacenamiento y posterior análisis.

#### **4.5.2. Procesamiento y Análisis de Datos**

El análisis de los datos se centró en la extracción de la temperatura foliar a partir de la matriz térmica y el cálculo del CWSI.

1. **Segmentación de la Imagen Térmica:** De cada matriz de 768 píxeles, se aplicó un algoritmo de segmentación para aislar los píxeles correspondientes exclusivamente a las hojas de la planta, descartando el fondo (suelo, maceta, estructura del invernadero). La temperatura foliar promedio ( $T_c$ ) se calculó a partir de estos píxeles segmentados.
2. **Cálculo del Índice de Estrés Hídrico (CWSI):** Se utilizó la metodología empírica del CWSI, desarrollada originalmente por Idso, Jackson y Reginato (Idso, Jackson, y Reginato, 1981), que normaliza la temperatura del dosel con respecto a las condiciones ambientales. El índice se calculó utilizando la siguiente fórmula (Quezada, Bastias, Quintana, Arancibia, y Solís, 2020):

$$CWSI = \frac{(T_c - T_{wet})}{(T_{dry} - T_{wet})}$$

Donde:

- $T_c$  es la temperatura del dosel medida por el sensor.

- $T_{wet}$  es la temperatura de referencia de una hoja que transpira libremente, calculada teóricamente a partir de la temperatura y humedad del aire.
- $T_{dry}$  es la temperatura de referencia de una hoja que no transpira, estimada como  $T_{ambiente} + 5^{\circ}\text{C}$ .

Un valor de CWSI cercano a 0 indica que la planta no tiene estrés, mientras que un valor cercano a 1 indica un estrés hídrico severo.

Los datos de temperatura foliar y CWSI se graficaron a lo largo del tiempo para visualizar las tendencias durante las tres fases del experimento.

## 4.6. Resultados

Los resultados obtenidos a lo largo de los 55 días de monitoreo demostraron la capacidad del sistema para detectar cambios en el estado hídrico de la planta de arándano.

Durante la fase de estrés hídrico inducido, se observó un incremento sostenido y claro en la temperatura foliar promedio, así como en el valor del CWSI. Este comportamiento es consistente con la respuesta fisiológica esperada de una planta que, al experimentar falta de agua, cierra sus estomas para reducir la pérdida por transpiración, lo que a su vez eleva su temperatura superficial (Jones, 2004).

Al reanudarse el riego en la fase de recuperación, el sistema registró un descenso rápido y significativo tanto en la temperatura foliar como en el CWSI. Este cambio abrupto indica que la planta reabrió sus estomas y reanudó la transpiración normal, enfriando sus hojas.

La correlación entre la suspensión del riego y el aumento de los valores térmicos, así como la rápida normalización de estos valores tras la rehidratación, constituye una prueba de concepto exitosa. Demuestra que el prototipo de bajo costo posee la sensibilidad necesaria para detectar

patrones térmicos directamente asociados al estrés hídrico. Si bien este experimento no incluyó un diseño estadístico con réplicas, los resultados son prometedores y validan el sistema como una herramienta viable para futuras investigaciones a mayor escala.

# V. Resultados y Conclusiones Finales

Este capítulo final presenta una síntesis de los resultados obtenidos a lo largo del proyecto, ofreciendo una discusión crítica sobre el cumplimiento de los objetivos, las implicaciones de los hallazgos y las futuras líneas de investigación. Se busca dar una respuesta clara a la pregunta de investigación y consolidar el aporte de este trabajo al campo de la agricultura de precisión.

## 5.1. Respuesta a la Pregunta de Investigación

La pregunta central que guio este proyecto fue: **¿Es posible desarrollar un sistema de bajo costo, basado en termografía infrarroja e inteligencia artificial, que permita detectar de manera temprana el estrés hídrico en plantas de arándano variedad Biloxi para optimizar la gestión del riego?**

La respuesta, a la luz de los resultados obtenidos, es afirmativa. El desarrollo y validación del sistema integrado por hardware, software y un estudio experimental han demostrado que es factible construir e implementar una solución tecnológica de bajo costo capaz de monitorear indicadores fisiológicos asociados al estrés hídrico.

- **Viabilidad Tecnológica:** Se diseñó, ensambló y validó metrológicamente un prototipo de hardware funcional utilizando componentes asequibles como el microcontrolador ESP32-S3 y el sensor térmico MLX90640. La caracterización del sensor demostró una precisión

aceptable para aplicaciones agrícolas, confirmando que la barrera del alto costo de los equipos comerciales puede ser superada.

- **Detección Temprana:** El estudio experimental, aunque concebido como una prueba de concepto, evidenció que el sistema es suficientemente sensible para registrar los cambios en la temperatura foliar de la planta de arándano en respuesta a la restricción de riego. El incremento sostenido del Índice de Estrés Hídrico del Cultivo (CWSI) durante la fase de sequía y su rápido descenso tras la rehidratación, constituye una prueba empírica de que el sistema detecta la respuesta fisiológica de la planta antes de que los signos de marchitamiento sean visualmente evidentes.

Por lo tanto, el proyecto demuestra exitosamente que la combinación de termografía de bajo costo y análisis de datos es una estrategia viable y prometedora para la detección temprana del estrés hídrico en el cultivo de arándano.

## 5.2. Discusión General

A continuación, se presenta un análisis crítico de la consecución de los objetivos planteados, las fortalezas y limitaciones del sistema, y su impacto potencial.

### 5.2.1. Consecución de los Objetivos Planteados

El proyecto cumplió satisfactoriamente con los objetivos específicos propuestos:

1. **Diseñar y construir un prototipo de hardware:** Se logró el diseño y ensamblaje de un dispositivo funcional, robusto y de bajo costo. La validación metrológica confirmó su fiabilidad como instrumento de medición, siendo este uno de los principales aportes del proyecto.

2. **Desarrollar un software de procesamiento:** Se implementó el firmware necesario para la adquisición continua de datos térmicos y ambientales. Adicionalmente, se desarrollaron los scripts para la segmentación de imágenes y el cálculo del CWSI, permitiendo transformar los datos crudos en un indicador agronómico valioso.
3. **Implementar un modelo de inteligencia artificial:** Aunque el estudio experimental se centró en la prueba de concepto y la validación del hardware, el diseño del sistema contempla la integración futura de modelos de IA. La recolección de un conjunto de datos etiquetados (planta con y sin estrés) durante 55 días sienta las bases para el entrenamiento de algoritmos de clasificación, cumpliendo con la fase inicial de este objetivo.
4. **Crear una aplicación web:** Se desarrolló una plataforma web que permite la visualización de los datos recopilados por el hardware, proporcionando una interfaz intuitiva para el usuario final. Esto completa el ciclo desde la captura del dato en campo hasta su presentación para la toma de decisiones.

### **5.2.2. Fortalezas y Limitaciones del Sistema**

#### **Fortalezas**

- **Bajo Costo:** La principal fortaleza del sistema es su asequibilidad. Al utilizar componentes de hardware de código abierto y ampliamente disponibles, el costo del prototipo es una fracción del de las cámaras termográficas comerciales, lo que democratiza el acceso a esta tecnología.
- **Monitoreo Continuo y No Invasivo:** A diferencia de los métodos tradicionales (e.g., bomba de Scholander), el sistema permite un monitoreo constante sin afectar a la planta, proporcionando una visión dinámica de su estado hídrico a lo largo del día y en diferentes condiciones climáticas.

- **Sistema Integral:** El proyecto no se limita al hardware, sino que ofrece una solución completa que abarca la captura de datos, el procesamiento, el almacenamiento y la visualización, lo que aumenta su valor práctico para el usuario final.

## Limitaciones

- **Alcance del Experimento:** La prueba de concepto se realizó con un solo individuo, lo que impide una validación estadística robusta y la generalización de los resultados. Factores como la variabilidad entre plantas y las diferentes condiciones microclimáticas de un cultivo real no fueron evaluados.
- **Resolución del Sensor Térmico:** La resolución de 32x24 píxeles del sensor MLX90640, aunque suficiente para esta prueba de concepto, puede ser una limitación para analizar plantas a mayor distancia o para obtener detalles finos de la distribución de temperatura en el dosel.
- **Dependencia de la Conectividad:** Aunque cuenta con almacenamiento local en una tarjeta SD, la funcionalidad en tiempo real del sistema depende de una conexión Wi-Fi estable para la transmisión de datos al servidor, lo cual puede ser un desafío en zonas rurales.

### 5.2.3. Impacto Potencial

La implementación de este sistema en un entorno real tiene el potencial de generar un impacto significativo en la agricultura del arándano. Al proporcionar a los agricultores una herramienta precisa y asequible para programar el riego basada en las necesidades reales de la planta, se pueden lograr beneficios como:

- **Optimización del Uso del Agua:** Reducción del consumo de agua al evitar el riego innecesario o excesivo.

- **Mejora del Rendimiento y Calidad del Cultivo:** Evitar el estrés hídrico, que afecta negativamente el tamaño y la calidad de la fruta.
- **Sostenibilidad Agrícola:** Promover prácticas agrícolas más sostenibles y resilientes al cambio climático.

### **5.3. Recomendaciones y Trabajo Futuro**

A partir de los resultados y las limitaciones identificadas, se proponen las siguientes líneas de trabajo futuro:

- **Escalamiento del Estudio Experimental:** Realizar experimentos con un mayor número de plantas, incluyendo réplicas y grupos de control. Evaluar el sistema en condiciones de campo reales para analizar su comportamiento frente a la variabilidad climática y del cultivo.
- **Integración de Inteligencia Artificial:** Utilizar el conjunto de datos recopilado para entrenar y validar modelos de aprendizaje automático (e.g., SVM, Redes Neuronales) que puedan clasificar automáticamente el estado hídrico de las plantas (e.g., "Normal", "Estrés Leve", "Estrés Severo") y generar alertas automáticas.
- **Mejora del Hardware:** Explorar la integración de sensores térmicos de mayor resolución a medida que su costo disminuya. Incorporar una cámara en el espectro visible para correlacionar los datos térmicos con imágenes RGB y aplicar técnicas de visión por computador para una mejor segmentación del dosel.
- **Calibración Específica del CWSI:** Desarrollar una línea base del CWSI específica para el arándano var. Biloxi en las condiciones climáticas de la sabana de Bogotá, lo que aumentaría la precisión del índice.

- **Desarrollo de una Red de Sensores:** Ampliar el sistema para crear una red de nodos de monitoreo distribuidos en un cultivo, permitiendo generar mapas de estrés hídrico y una gestión del riego por zonas.

## 5.4. Conclusiones Finales

Este trabajo de grado ha cumplido exitosamente su objetivo principal al demostrar que es posible desarrollar un sistema funcional y de bajo costo para la detección de estrés hídrico en plantas de arándano Biloxi mediante termografía infrarroja.

Se ha diseñado y validado un prototipo de hardware que representa una alternativa asequible a los equipos comerciales. A través de un estudio experimental controlado, se comprobó que el sistema es capaz de detectar los cambios fisiológicos en la planta asociados a la falta de agua, validando la prueba de concepto.

El proyecto no solo aporta una solución tecnológica, sino que también contribuye con un valioso conjunto de datos de 55 días de monitoreo continuo, que servirá como base para el desarrollo de futuros modelos de inteligencia artificial. A pesar de sus limitaciones, este trabajo sienta las bases para futuras investigaciones y desarrollos que podrían tener un impacto positivo en la sostenibilidad y eficiencia de la producción de arándanos en Colombia, facilitando la adopción de prácticas de agricultura de precisión en un sector clave de la economía nacional.

## Referencias

- Bosch Sensortec. (2018). *Bst-bme280-ds002* (Inf. Téc.). Bosch Sensortec GmbH. (Revisión 1.14)
- Congreso de la República de Colombia. (2009). *Ley 1273 de 2009. por medio de la cual se modifica el código penal, se crea un nuevo bien jurídico tutelado - denominado "de la protección de la información y de los datosz se preservan integralmente los sistemas que utilicen las tecnologías de la información y las comunicaciones, entre otras disposiciones.* Bogotá: Diario Oficial No. 47.223.
- Congreso de la República de Colombia. (2012). *Ley estatutaria 1581 de 2012. por la cual se dictan disposiciones generales para la protección de datos personales.* Bogotá: Diario Oficial No. 48.587.
- Consejo Superior de la Universidad de Cundinamarca. (2018). *Acuerdo no. 000004. por medio del cual se adopta el estatuto de propiedad intelectual de la universidad de cundinamarca.* Fusagasugá: Universidad de Cundinamarca.
- Idso, S. B., Jackson, R. D., y Reginato, R. J. (1981). Remote sensing of crop yields. *Science*, 214(4516), 19–25. doi: 10.1126/science.7025257
- JCGM. (2008). *International vocabulary of metrology – basic and general concepts and associated terms (vim).* Joint Committee for Guides in Metrology. (JCGM 200:2008)
- Jones, H. G. (2004). Application of thermal imaging and infrared sensing in plant physiology and ecophysiology. *Advances in Botanical Research*, 41, 107–163. doi: 10.1016/S0065-2296(04)41003-9
- Melexis. (2021). *Mlx90640 datasheet: 32x24 ir array* (Inf. Téc.). Melexis. (Revisión 12)
- Poblete-Echeverría, C., Fuentes, S., Ortega-Farias, S., Gonzalez-Talice, J., y Yuri, J. A. (2023). Infrared thermal imaging for detecting water stress in blueberry plants. *Journal of Agronomy and Crop Science*, 209(1), 56–70. doi: 10.1111/jac.12582

- Quezada, C., Bastias, R., Quintana, R., Arancibia, R., y Solís, A. (2020). Validación del Índice de estrés hídrico de cultivo (cwsí) mediante termografía infrarroja y su incidencia en rendimiento y calidad en manzanas ‘royal gala’. *Chilean Journal of Agricultural and Animal Sciences*, 36(3), 200–208. doi: 10.29393/CHJAAS36-18VICQ50018
- Rinza, J., Ramírez, D. A., y Ninanya, J. (2021). *Guía de práctica de campo: Monitoreo de rasgos funcionales en los cultivos para la detección de estrés temprano: La conductancia estomática y termografía infrarroja como herramienta de medidas claves* (Inf. Téc.). Centro Internacional de la Papa. doi: 10.4160/9789290605676
- Rodríguez-Alonso, D., y Cabrejo-Paredes, J. (2022). Excelente confiabilidad de la cámara termográfica de bolsillo para apoyar el diagnóstico de la neuropatía periférica diabética en atención primaria. *Revista Médica Vallejiana*, 11(2), 11–20. doi: 10.18050/revistamedicavallejiana.v11i2.01
- Rumbaugh, J., Jacobson, I., y Booch, G. (2007). *El lenguaje unificado de modelado: Manual de referencia* (2a ed.). Madrid: Pearson Addison Wesley.
- Salgado Vargas, C., Sánchez-García, P., Volke-Haller, V. H., y Colinas León, M. T. B. (2018). Respuesta agronómica de arándano (*vaccinium corymbosum* L.) al estrés osmótico. *Agrociencia*, 52(2), 231–239. Descargado de [https://www.scielo.org.mx/scielo.php?script=sci\\_arttext&pid=S1405-31952018000200231](https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-31952018000200231)