

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERÍA**

**Escuela Profesional de Ingeniería de Sistemas**

**Aplicativo Móvil para Identificar Enfermedades de la Hoja de Orégano mediante Deep Learning en la región de Tacna**

**Curso:** Construcción de Software I

**Docente:** Ing. Alberto Johnatan Flor Rodriguez

**Integrantes:**

Cuadros Napa, Raúl Marcelo (2017057851)

Melendez Huarachi, Gabriel Fari (2021070311)

**Tacna – Perú**

***2025***

| CONTROL DE VERSIONES | | | | | |
| --- | --- | --- | --- | --- | --- |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | GFM, RMC | AJF | AJF | 14/04/2025 | Avance de la primera versión del documento |
| 2.0 | GFM, RMC |  |  | 16/05/2025 | Avance del segundo entregable del documento |

**Aplicativo Móvil para Identificar Enfermedades de la Hoja de Orégano mediante Deep Learning en la región de Tacna**

**Documento de Arquitectura de Software**

**Versión *1.0***

**ÍNDICE GENERAL**

[***1.***](#_heading=h.1fob9te) ***INTRODUCCIÓN* 4**

[**1.1.**](#_heading=h.3znysh7) **Propósito (Diagrama 4+1)** 4

[**1.2.**](#_heading=h.2et92p0) **Alcance** 4

[**1.3.**](#_heading=h.tyjcwt) **Definición, siglas y abreviaturas** 4

[**1.4.**](#_heading=h.3dy6vkm) **Organización del documento 5**

[***2.***](#_heading=h.1t3h5sf) ***OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS 5***

[2.1.1.](#_heading=h.4d34og8) Requerimientos Funcionales 6

[2.1.2.](#_heading=h.17dp8vu) Requerimientos No Funcionales – Atributos de Calidad 10

[***3.***](#_heading=h.3rdcrjn) ***REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA* 12**

[**3.1.**](#_heading=h.lnxbz9) **Vista de Caso de uso** 12

[3.1.1.](#_heading=h.35nkun2) Diagramas de Casos de uso 12

[**3.2.**](#_heading=h.1ksv4uv) **Vista Lógica 19**

[3.2.1.](#_heading=h.2jxsxqh) Diagrama de Subsistemas (paquetes) 19

[3.2.2.](#_heading=h.z337ya) Diagrama de Secuencia (vista de diseño) 19

[3.2.3.](#_heading=h.3j2qqm3) Diagrama de Colaboración (vista de diseño) 25

[3.2.4.](#_heading=h.1y810tw) Diagrama de Objetos 25

[3.2.5.](#_heading=h.4i7ojhp) Diagrama de Clases 34

[3.2.6.](#_heading=h.2xcytpi) Diagrama de Base de datos (relacional o no relacional) 35

[**3.3.**](#_heading=h.1ci93xb) **Vista de Implementación (vista de desarrollo) 36**

[3.3.1.](#_heading=h.3whwml4) Diagrama de arquitectura software (paquetes) 37

[3.3.2.](#_heading=h.2bn6wsx) Diagrama de arquitectura del sistema (Diagrama de componentes) 37

[**3.4.**](#_heading=h.qsh70q) **Vista de procesos** 37

[3.4.1.](#_heading=h.3as4poj) Diagrama de Procesos del sistema (diagrama de actividad) 37

[**3.5.**](#_heading=h.1pxezwc) **Vista de Despliegue (vista física) 39**

[3.5.1.](#_heading=h.49x2ik5) Diagrama de despliegue 39

[***4.***](#_heading=h.2p2csry) ***ATRIBUTOS DE CALIDAD DEL SOFTWARE* 39**

[**Escenario de Funcionalidad**](#_heading=h.147n2zr) 40

[**Escenario de Usabilidad**](#_heading=h.3o7alnk) 40

[**Escenario de confiabilidad**](#_heading=h.23ckvvd) 41

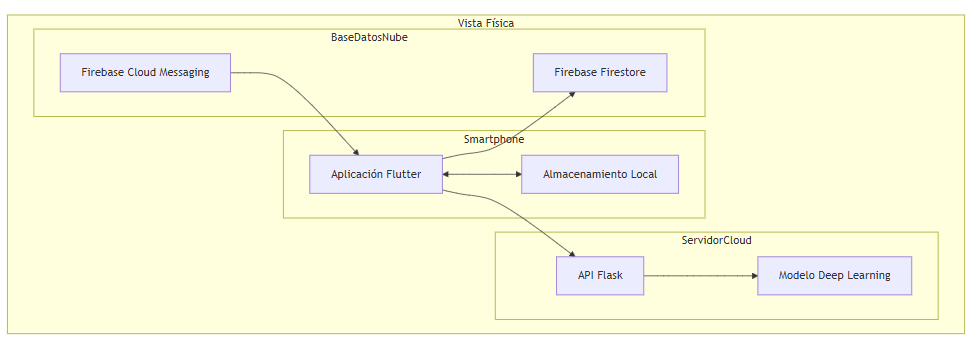
[**Escenario de rendimiento**](#_heading=h.ihv636) 41

[**Escenario de mantenibilidad**](#_heading=h.32hioqz) 42

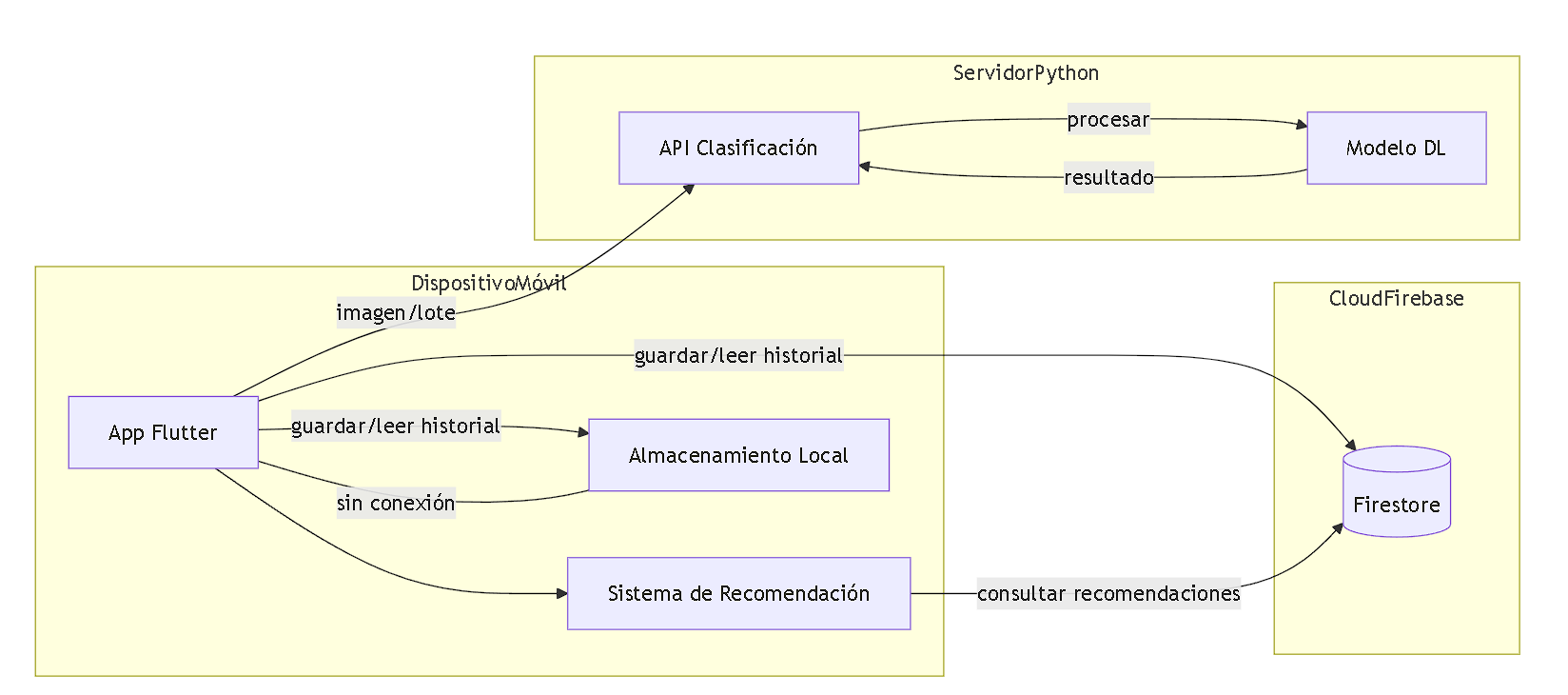
[**Otros Escenarios**](#_heading=h.1hmsyys) 42

1. **INTRODUCCIÓN**
   1. **Propósito (Diagrama 4+1)**

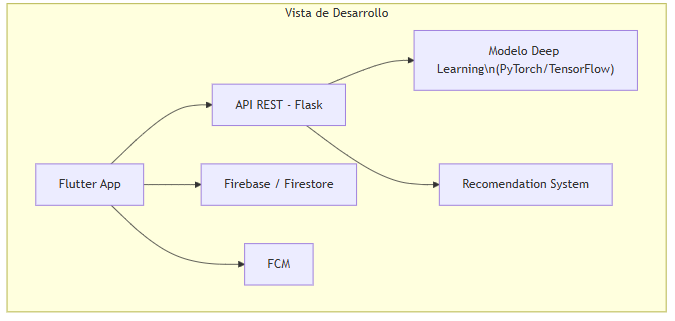
El propósito del presente proyecto es desarrollar un aplicativo móvil inteligente que permita a los agricultores y técnicos agrícolas de la región de Tacna identificar de manera automática las enfermedades en las hojas de la planta de orégano, mediante el uso de técnicas de Deep Learning aplicadas al análisis de imágenes.



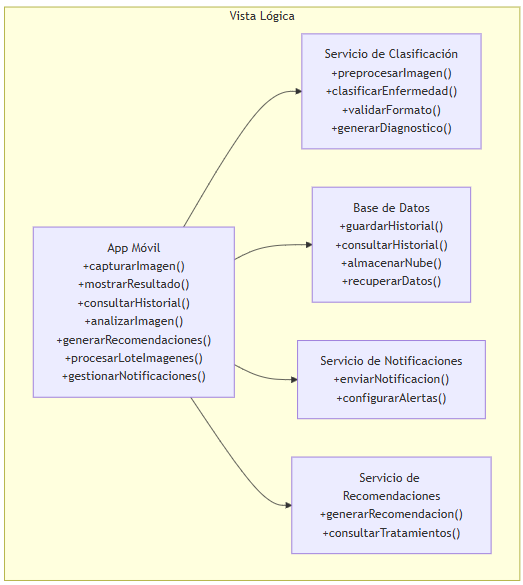
*Figura N°1: Arquitectura de Vista Física*



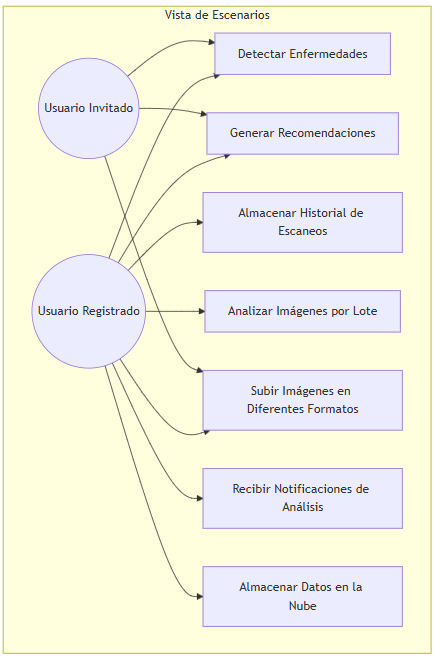
*Figura N°2: Arquitectura de Vista de Funcionamiento*



*Figura N°3: Arquitectura de Vista de Desarrollo*



*Figura N°4: Arquitectura de Vista de Logica*



*Figura N°5: Arquitectura de Vista de Escenarios*

* 1. **Alcance**

El proyecto abarca el desarrollo e implementación de una aplicación móvil que utiliza Deep Learning para detectar enfermedades en las hojas de orégano en la región de Tacna, permitiendo a los agricultores obtener diagnósticos en tiempo real a través del análisis de imágenes. Incluye el entrenamiento de un modelo de inteligencia artificial, el desarrollo de una API en Python (Flask), la integración con Firebase para almacenamiento de datos y la implementación en dispositivos móviles con Flutter. Además, se contempla la provisión de recomendaciones para el manejo de enfermedades y el despliegue en un servidor o en un Single Board Computer (SBC), asegurando su funcionalidad tanto en línea como fuera de línea.

* 1. **Definición, siglas y abreviaturas**
* Deep Learning (DL): Subcampo del aprendizaje automático basado en redes neuronales profundas para el procesamiento y análisis de datos complejos.
* Inteligencia Artificial (IA): Rama de la informática que desarrolla sistemas capaces de realizar tareas que requieren inteligencia humana, como el reconocimiento de imágenes y la toma de decisiones.
* Red Neuronal Convolucional (CNN): Tipo de red neuronal utilizada en visión por computadora para analizar imágenes y extraer patrones.
* Flutter: Framework de desarrollo de aplicaciones móviles multiplataforma creado por Google.
* Firebase: Plataforma de Google para el desarrollo de aplicaciones con servicios en la nube, como bases de datos y autenticación.
* Python: Lenguaje de programación utilizado en el desarrollo del modelo de deep learning y la API del sistema.
* Flask: Framework ligero de Python para el desarrollo de aplicaciones web y APIs.
* API (Application Programming Interface): Conjunto de funciones y protocolos que permiten la comunicación entre aplicaciones.
* SBC (Single Board Computer): Computadora de placa única que integra todos los componentes esenciales en un solo circuito.
* TensorFlow Lite: Versión optimizada de TensorFlow para la ejecución de modelos de machine learning en dispositivos móviles y embebidos.
* GPU (Graphics Processing Unit): Unidad de procesamiento gráfico utilizada para acelerar el entrenamiento de modelos de deep learning.
* Git: Sistema de control de versiones que permite gestionar y colaborar en proyectos de software.
* GitHub: Plataforma basada en Git para almacenamiento, control de versiones y colaboración en desarrollo de software.

1. **OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS**
   1. **Priorización de requerimientos**

| **Código** | **Requerimientos**  **Funcionales** | **Descripción** | **Prioridad** | **Importancia** |
| --- | --- | --- | --- | --- |
| RF01 | Detectar Enfermedades | El sistema debe analizar imágenes y detectar enfermedades utilizando algoritmos avanzados de procesamiento de imágenes. | 3 | Alta |
| RF02 | Generar recomendaciones | El sistema debe generar recomendaciones basadas en los resultados del análisis de las imágenes, como tratamientos o cambios de estilo de vida. | 2 | Media |
| RF03 | Almacenar historial de escaneos | El sistema debe permitir almacenar y consultar el historial de escaneos realizados por el usuario. | 2 | Media |
| RF04 | Analizar Imágenes por Lote | El sistema debe permitir cargar y analizar múltiples imágenes de una vez (Máximo 50 imágenes). | 1 | Baja |
| RF05 | Subir Imagenes en Diferentes Formatos | El sistema debe aceptar imágenes en diversos formatos como JPG, PNG, JPEG, BMP, WEBP. | 2 | Media |
| RF06 | Revisar Notificación de Análisis | El sistema debe permitir revisar notificaciones sobre los resultados del análisis, como diagnósticos y recomendaciones. | 3 | Alta |
| RF07 | Almacenar Datos en la Nube | El sistema debe almacenar datos en la nube para acceder a ellos desde cualquier dispositivo de forma segura. | 2 | Media |

*Tabla N°1: Tabla de Requerimientos funcionales inicial*

**2.2 Requerimientos Funcionales**

| **Código** | **Requerimientos**  **Funcionales** | **Descripción** | **Prioridad** | **Importancia** |
| --- | --- | --- | --- | --- |
| RF01 | Detectar Enfermedades | El sistema debe analizar imágenes y detectar enfermedades utilizando algoritmos avanzados de procesamiento de imágenes. | 3 | Alta |
| RF02 | Generar recomendaciones | El sistema debe generar recomendaciones basadas en los resultados del análisis de las imágenes, como tratamientos o cambios de estilo de vida. | 2 | Media |
| RF03 | Almacenar historial de escaneos | El sistema debe permitir almacenar y consultar el historial de escaneos realizados por el usuario. | 2 | Media |
| RF04 | Analizar Imágenes por Lote | El sistema debe permitir cargar y analizar múltiples imágenes de una vez. | 1 | Baja |
| RF05 | Subir Imagenes en Diferentes Formatos | El sistema debe aceptar imágenes en diversos formatos como JPG, PNG, GIF. | 2 | Media |
| RF06 | Revisar Notificación de Análisis | El sistema debe permitir revisar notificaciones sobre los resultados del análisis, como diagnósticos y recomendaciones. | 3 | Alta |
| RF07 | Almacenar Datos en la Nube | El sistema debe almacenar datos en la nube para acceder a ellos desde cualquier dispositivo de forma segura. | 2 | Media |

*Tabla N°2: Cuadro de requerimientos funcionales final*

**2.3 Requerimientos No Funcionales – Atributos de Calidad**

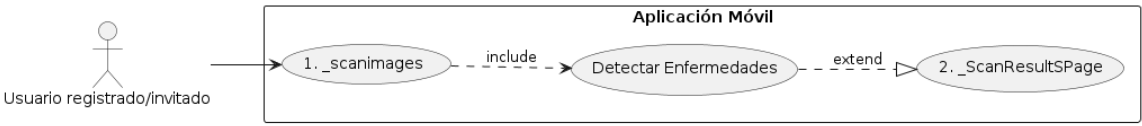
| **Código** | **Requerimientos No**  **Funcionales** | **Descripción** | **Prioridad** | **Importancia** |
| --- | --- | --- | --- | --- |
| RNF01 | Usabilidad | La aplicación debe ser intuitiva y fácil de usar para agricultores, considerando que muchos pueden tener conocimientos tecnológicos limitados. | 3 | Alta |
| RNF02 | Rendimiento | La solución debe procesar imágenes y entregar resultados en tiempo real o casi real. | 2 | Media |
| RNF03 | Compatibilidad | La aplicación debe ser compatible con los principales sistemas operativos móviles. | 2 | Media |

*Tabla N°3: Cuadro de Requerimientos no funcionales*

1. **REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA**

* 1. **Vista de Caso de uso**

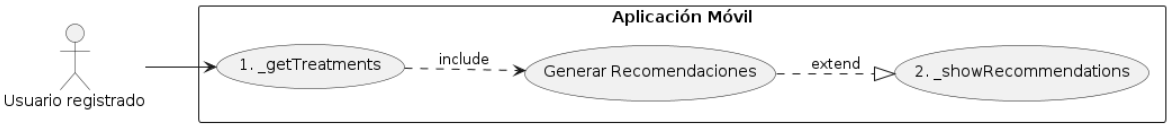
1. **Caso de Uso 01 - Detectar Enfermedades:**



*Figura N°6: Caso de Uso 01 - Detectar Enfermedades*

*Nota: Este diagrama muestra cómo los usuarios (registrados o invitados) usan una aplicación móvil para diagnosticar enfermedades en cultivos: primero capturan imágenes de las hojas \_scanImages, luego el sistema las analiza automáticamente Detectar Enfermedades, y finalmente muestra los resultados ScanResultsPage.*

1. **Caso de Uso 02 - Generar Recomendaciones:**

****

*Figura N°7: Caso de Uso 02 - Generar Recomendaciones*

*Nota: La aplicación móvil permite a usuarios registrados obtener recomendaciones agrícolas mediante un flujo automatizado: inicia con la búsqueda de tratamientos \_getTreatments, genera recomendaciones personalizadas Generar Recomendaciones y finalmente muestra los resultados \_showRecommendations.*

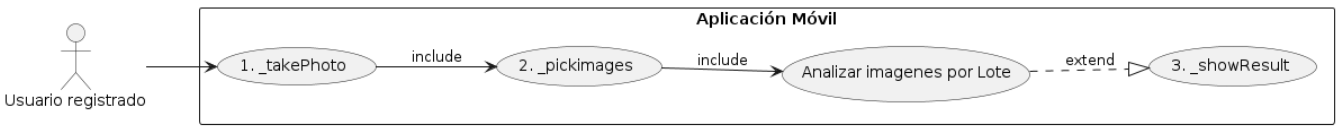
1. **Caso de Uso 03 - Almacenar Historial de Escaneos**

****

*Figura N°8: Caso de Uso 03 - Almacenar Historial de Escaneos*

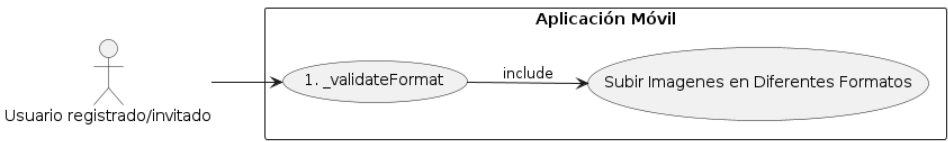
*Nota: La aplicación móvil permite a usuarios registrados gestionar diagnósticos agrícolas mediante \_registerDate donde va almacenar los datos y classifyDisease donde analiza enfermedades y saveImage donde guarda las imágenes.*

1. **Caso de Uso 04 - Analizar Imágenes por Lote**

*****Figura N°9: Caso de Uso 04 - Analizar Imágenes por Lote*

*Nota: Este diagrama describe el análisis de imágenes para usuarios registrados. Inicia con \_takePhoto, donde se captura fotos. Estas imágenes pasan a \_pickImages para su selección. Luego se realiza un Análisis por Lote, el cual va mostrar los resultados en \_showResult.*

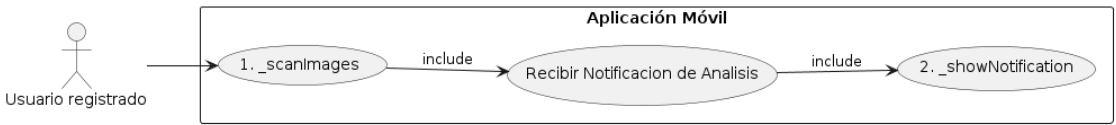
1. **Caso de Uso 05 - Subir Imágenes en Diferentes Formatos**

****

*Figura N°10: Caso de Uso 05 - Subir Imágenes en Diferentes Formatos*

*Nota: Este diagrama muestra el proceso de validación de imágenes en la aplicación móvil para usuarios (registrados o invitados). Comienza con \_validateFormat, que verifica el formato de las imágenes y luego pasa recién a Subir Imagenes en Diferentes Formatos para cargarlas al aplicativo móvil.*

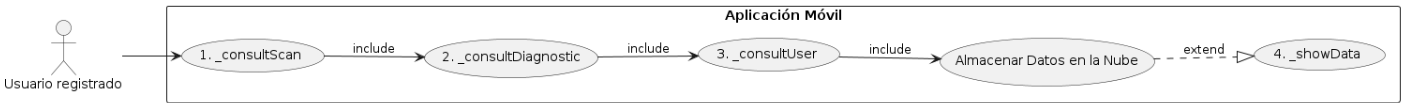
1. **Caso de Uso 06 - Recibir Notificación de Análisis**

****

*Figura N°11: Caso de Uso 06 - Recibir Notificación de Análisis*

*Nota: Este diagrama describe las notificaciones en la aplicación móvil para usuarios registrados. El proceso inicia con \_scanImages), que se requiere obligatoriamente. Luego se Recibe Notificación de Análisis para informar al usuario sobre el progreso, y culmina con \_showNotification para visualizar los resultados.*

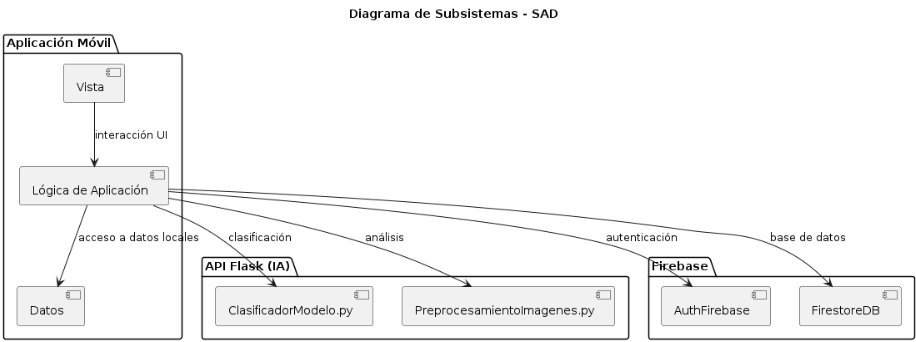
1. **Caso de Uso 07 - Almacenar Datos en la Nube**



*Figura N°12: Caso de Uso 07 - Almacenar Datos en la Nube*

*Nota: Este diagrama describe el proceso de diagnóstico agrícola para usuarios registrados, donde primero se realiza el escaneo \_consulScan, luego el análisis \_consulDiagnostic, seguido de la validación de usuario \_consulUser, y finalmente Almacena Datos en la Nube. La visualización de resultados se muestra en \_showData.*

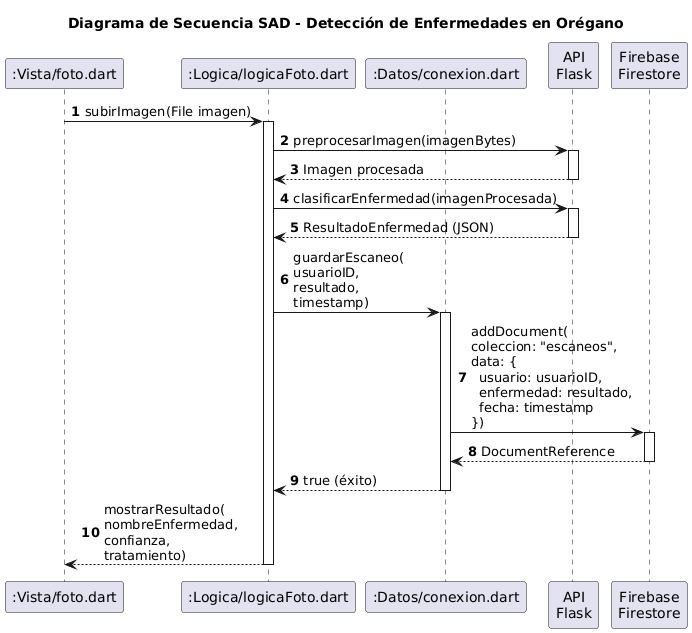
* 1. Vista Lógica
     1. Diagrama de Subsistemas (paquetes)



*Figura N°13: Arquitectura de Vista de Desarrollo*

* + 1. Diagrama de Secuencia (vista de diseño)

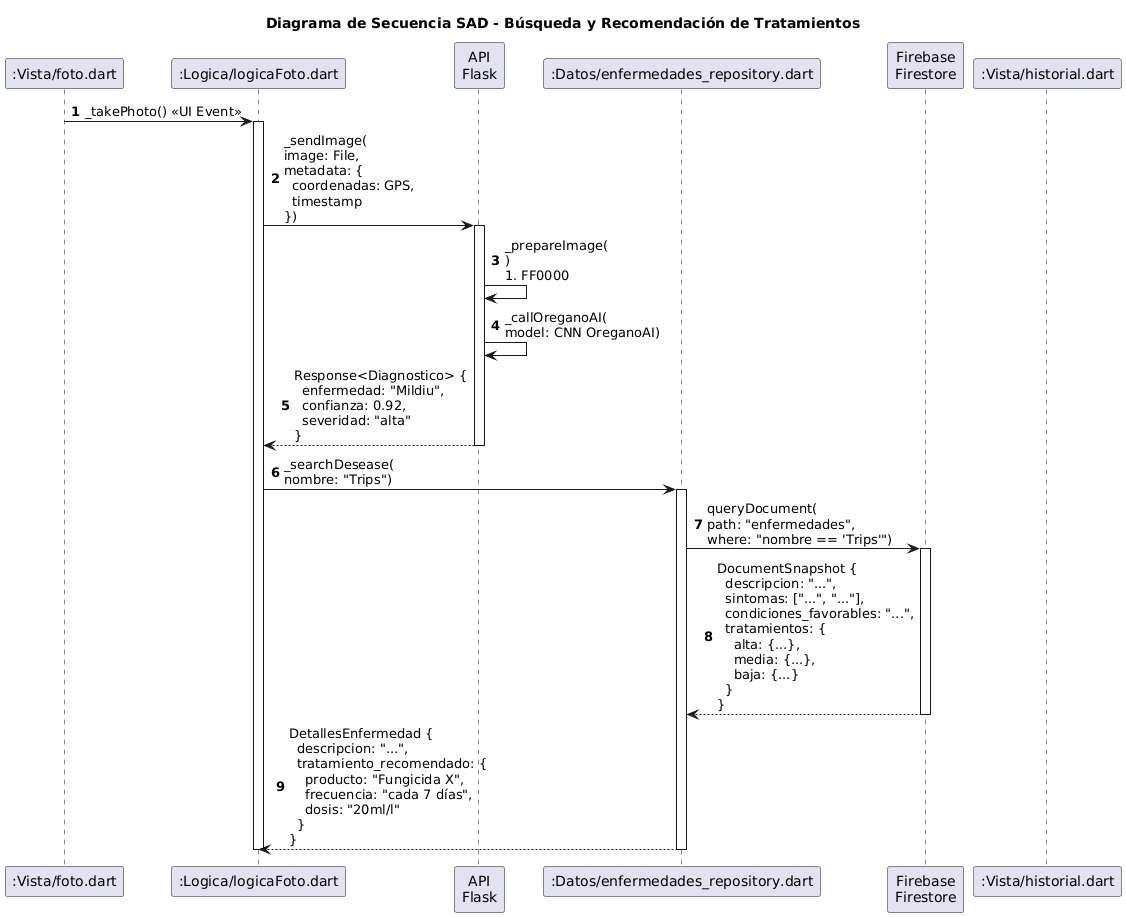
**Diagrama de Secuencia 01 - Detectar Enfermedades:**



*Figura N°14: Diagrama de Secuencia 01 - Detectar Enfermedades*

*Nota: Este diagrama muestra el flujo básico donde un usuario sube una imagen de una hoja para su análisis. El sistema procesa la imagen mediante un modelo de IA, determina si hay enfermedad y muestra el resultado.*

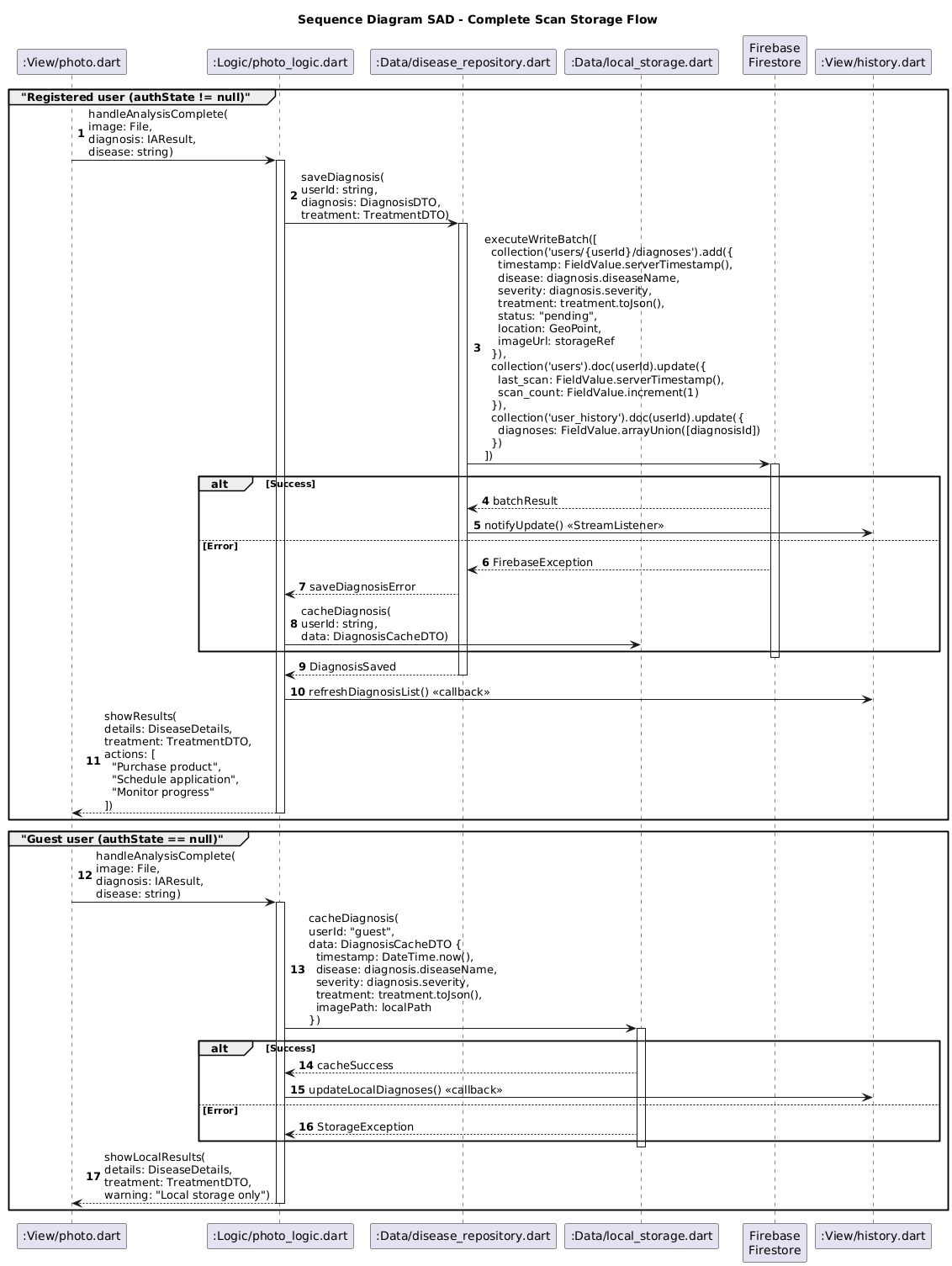
1. **Diagrama de Secuencia 02 - Generar Recomendaciones:**

****

*Figura N°15: Diagrama de Secuencia 02 - Generar Recomendaciones*

*Nota: Este diagrama muestra cómo el agricultor fotografía su cultivo, el sistema detecta enfermedades (ej: Mildiu) mediante IA, sugiere tratamientos específicos (como "aplicar fungicida cada 7 días") y guarda las recomendaciones en el historial para acciones rápidas y seguimiento futuro.*

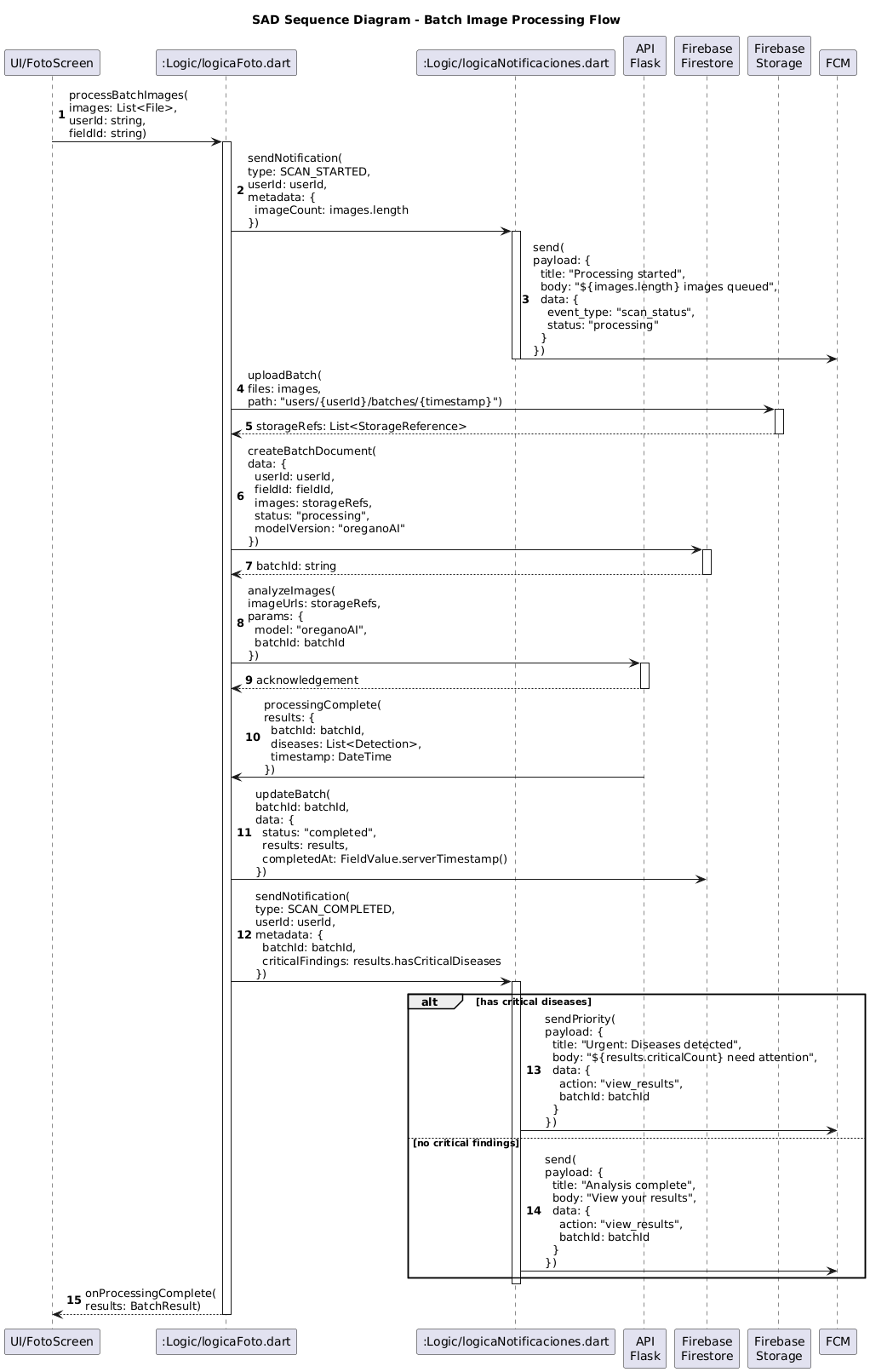
1. **Diagrama de Secuencia 03 - Almacenar Historial de Escaneos:**

****

*Figura N°16: Diagrama de Secuencia 03 - Almacenar Historial de Escaneos*

*Nota: Este diagrama detalla cómo el sistema guarda automáticamente cada análisis (imagen, diagnóstico, fecha y enfermedad) en el historial cuando el usuario está registrado.*

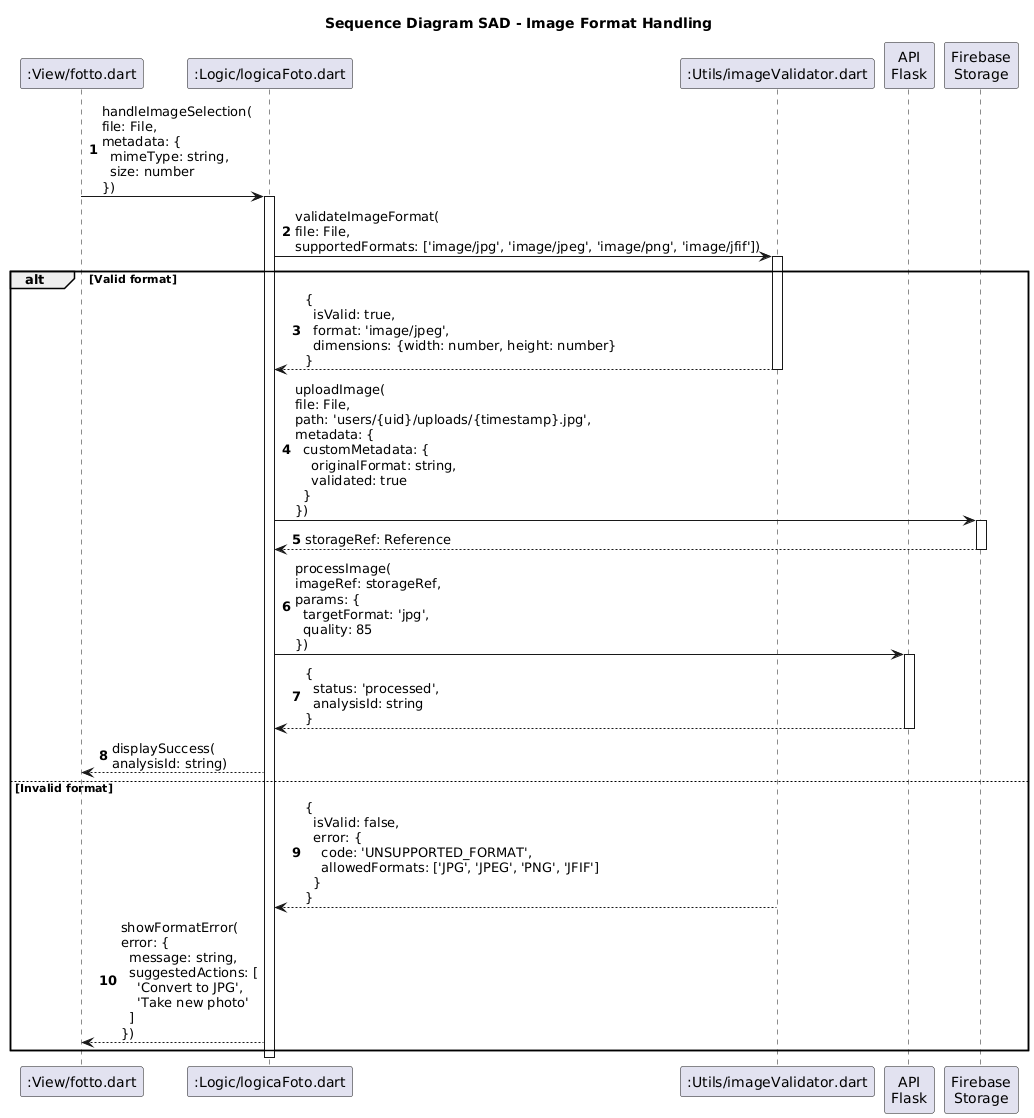
1. **Diagrama de Secuencia 04 - Analizar Imágenes por Lote:**

****

*Figura N°17: Diagrama de Secuencia 04 - Analizar Imágenes por Lote*

*Nota: Aquí se ilustra el procesamiento eficiente de múltiples imágenes en un solo lote.*

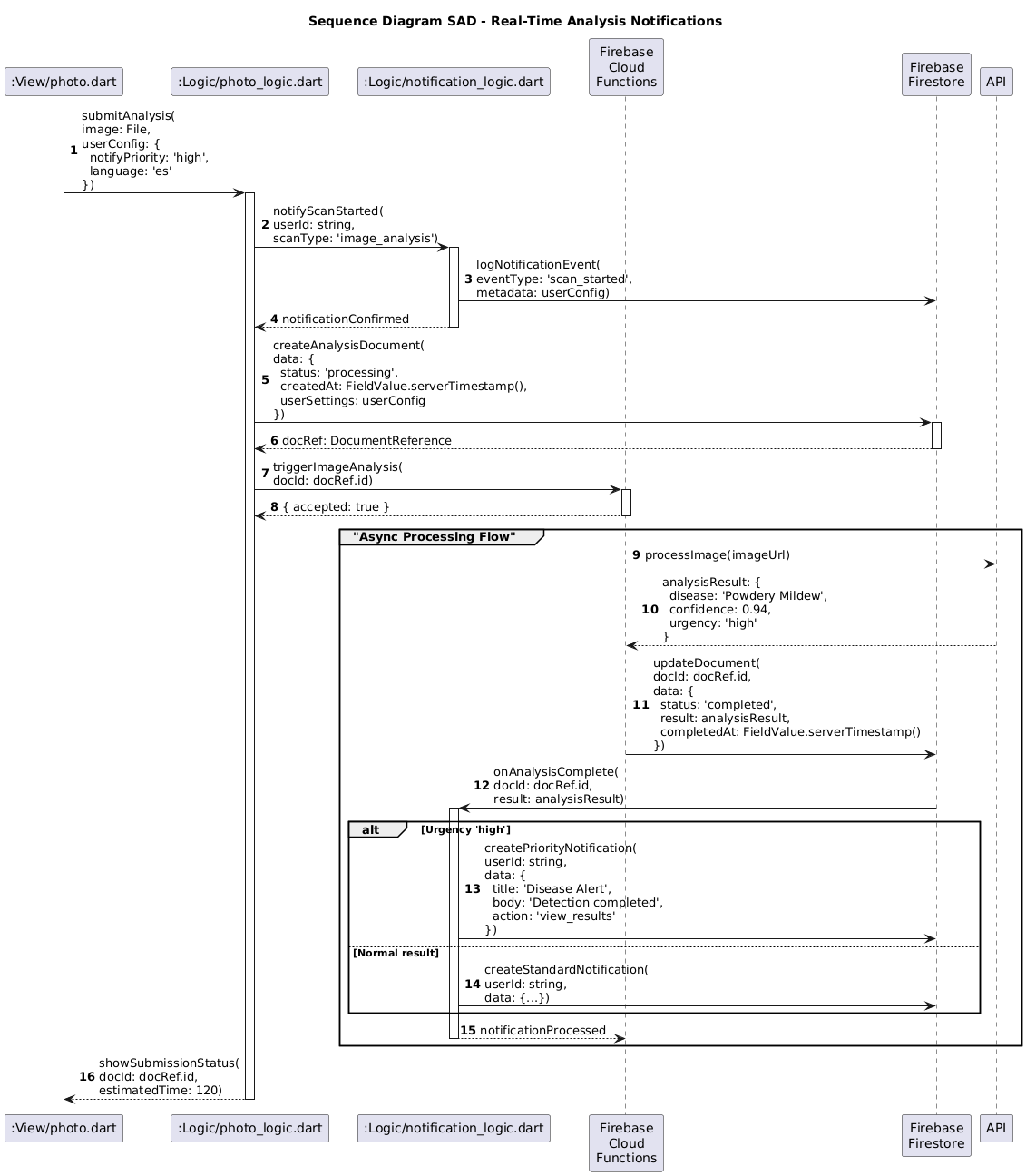
1. **Diagrama de Secuencia 05 - Subir Imágenes en Diferentes Formatos:**

****

*Figura N°18: Diagrama de Secuencia 05 - Subir Imágenes en Diferentes Formatos*

*Nota: El diagrama ilustra el proceso donde el usuario sube una imagen de su cultivo (JPG/PNG/GIF), el sistema verifica si el formato es válido, lo procesa si es compatible o muestra un mensaje de error claro si no lo es, asegurando una experiencia fluida al cargar imágenes sin problemas técnicos.*

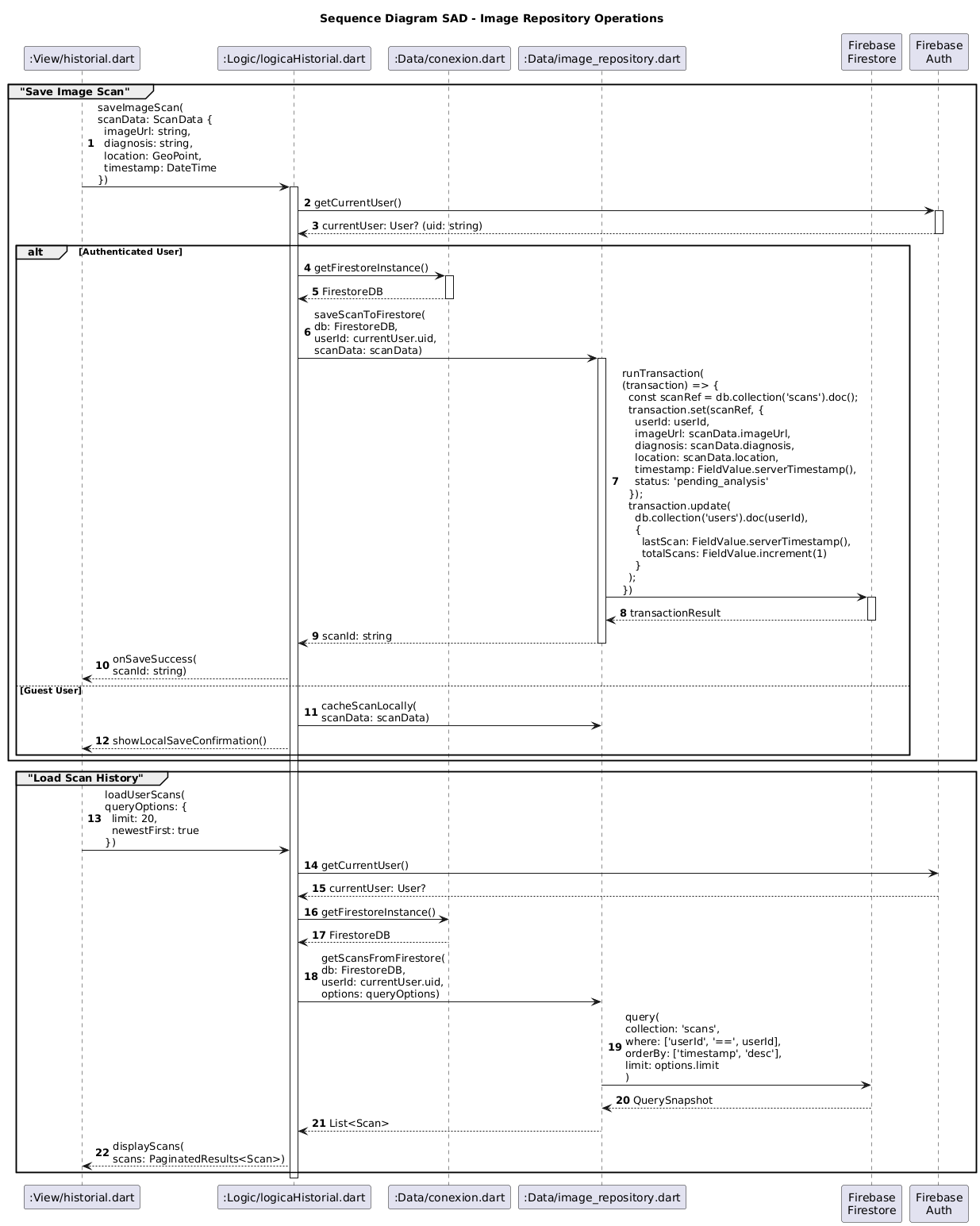
1. **Diagrama de Secuencia 06 - Recibir Notificación de Análisis:**

**

*Figura N°19: Diagrama de Secuencia 06 - Recibir Notificaciones de Análisis*

*Nota: Este flujo muestra cómo el sistema notifica al usuario en tiempo real sobre los resultados de sus análisis. Si se detecta una enfermedad, se envía una alerta urgente.*

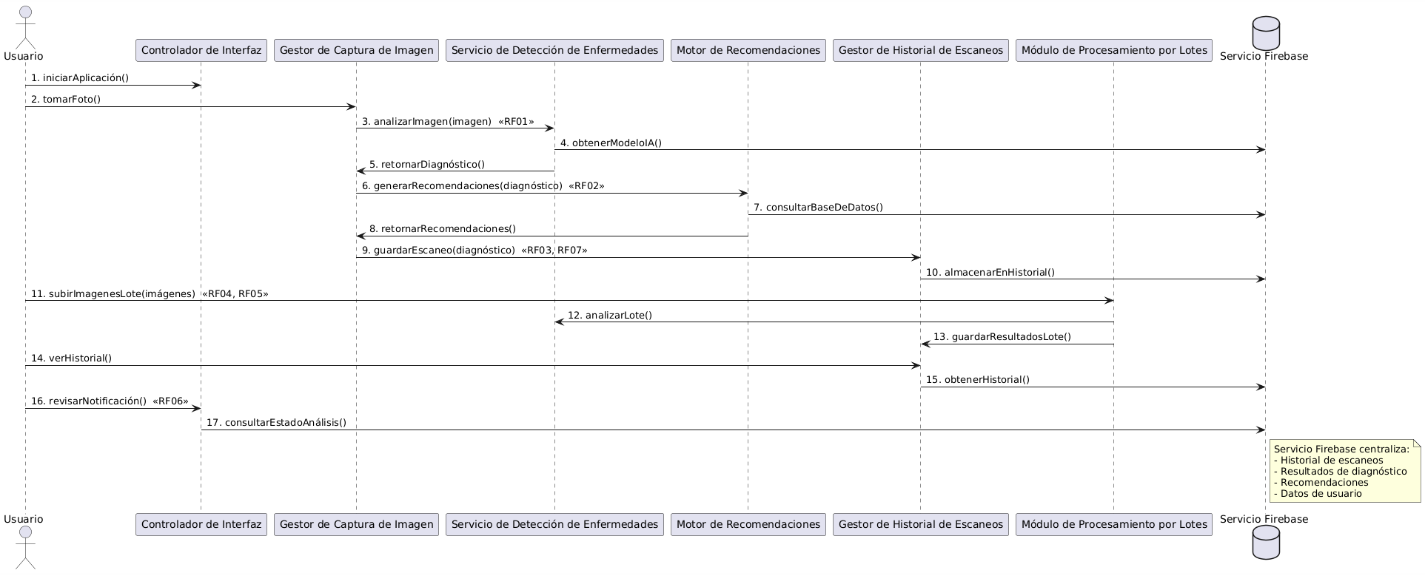
1. **Diagrama de Secuencia 07 - Almacenar Datos en la Nube:**

****

*Figura N°20: Diagrama de Secuencia 07 - Almacenar Datos en la Nube*

*Nota: Este flujo describe cómo la app guarda los escaneos e información del usuario en una base de datos en la nube, permitiendo acceder al historial desde cualquier dispositivo de forma instantánea, lo que facilita el monitoreo continuo de los cultivos sin depender de un solo equipo.*

* + 1. Diagrama de Colaboración (vista de diseño)



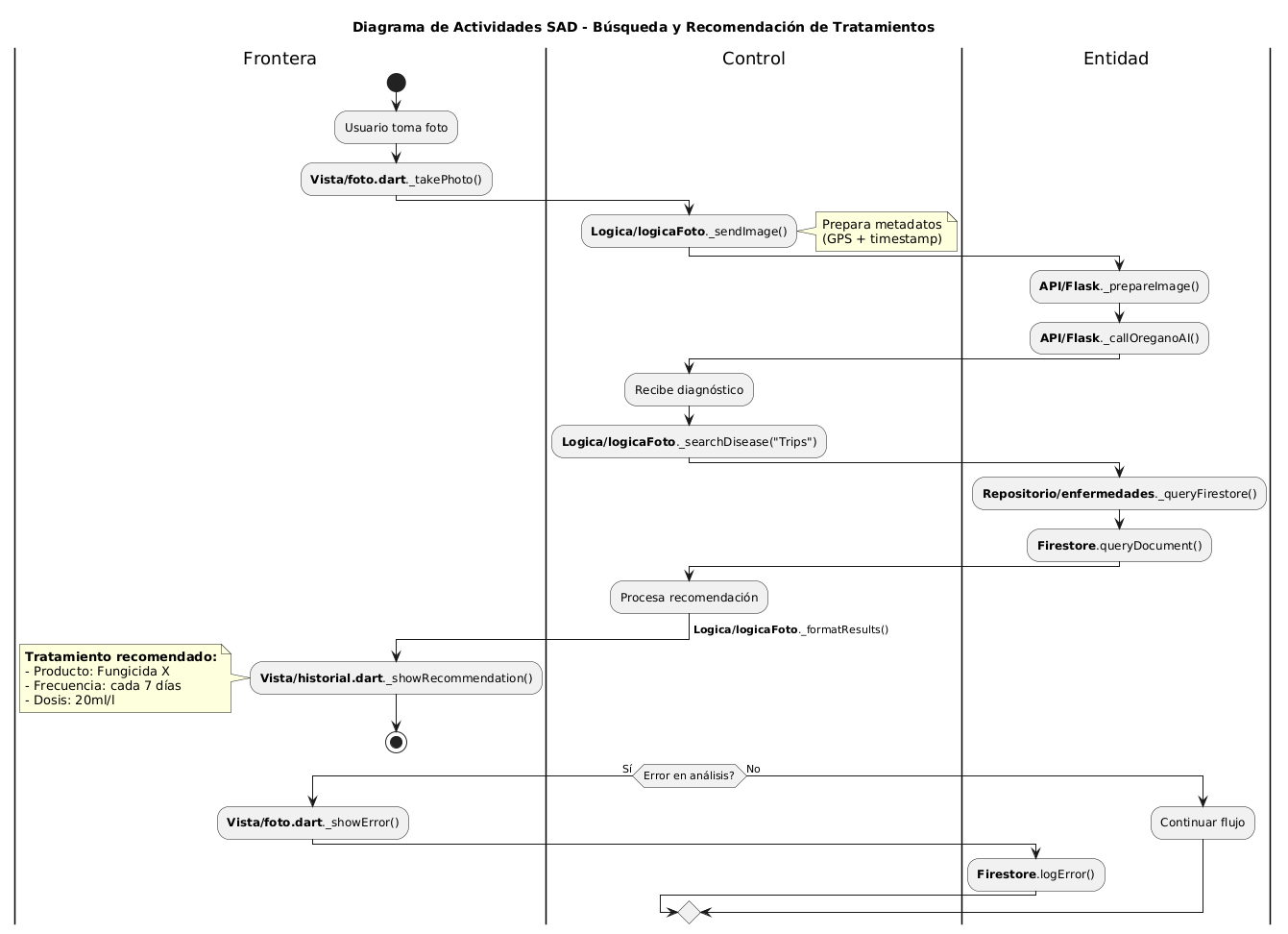
*Figura N°21: Diagrama de Colaboración*

* + 1. Diagrama de Objetos

| FRONTERA | CONTROL | ENTIDAD |
| --- | --- | --- |
| login | logicaLogin | AuthFirebase |
| register | logicaRegister | FirestoreDB |
| configuracion | logicaSesion | conexion |
| perfil | logicaFoto | escaneos\_memoria |
| foto | logicaNotificaciones | apiModeloIA |
| historial | firebaseAuth |  |

**b) Diagrama de Actividades con objetos**

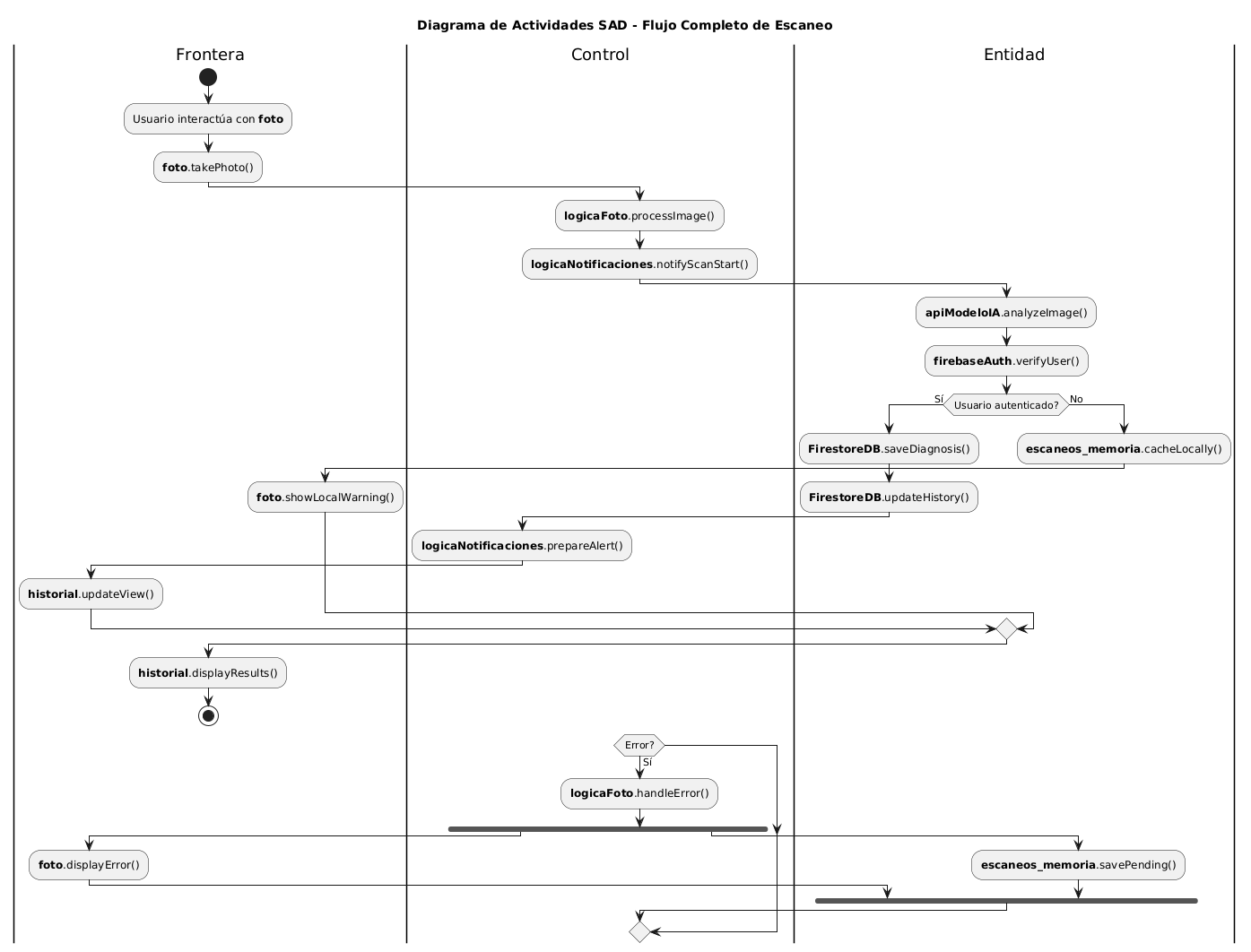
*Diagrama de Actividades con objetos 02 - Generar Recomendaciones:*

**

*Figura N°22: Diagrama de Actividades con objetos 02 - Generar Recomendaciones*

*Nota: Este diagrama muestra cómo el agricultor fotografía su cultivo, el sistema detecta enfermedades (ej: Mildiu) mediante IA, sugiere tratamientos específicos (como "aplicar fungicida cada 7 días") y guarda las recomendaciones en el historial para acciones rápidas y seguimiento futuro.*

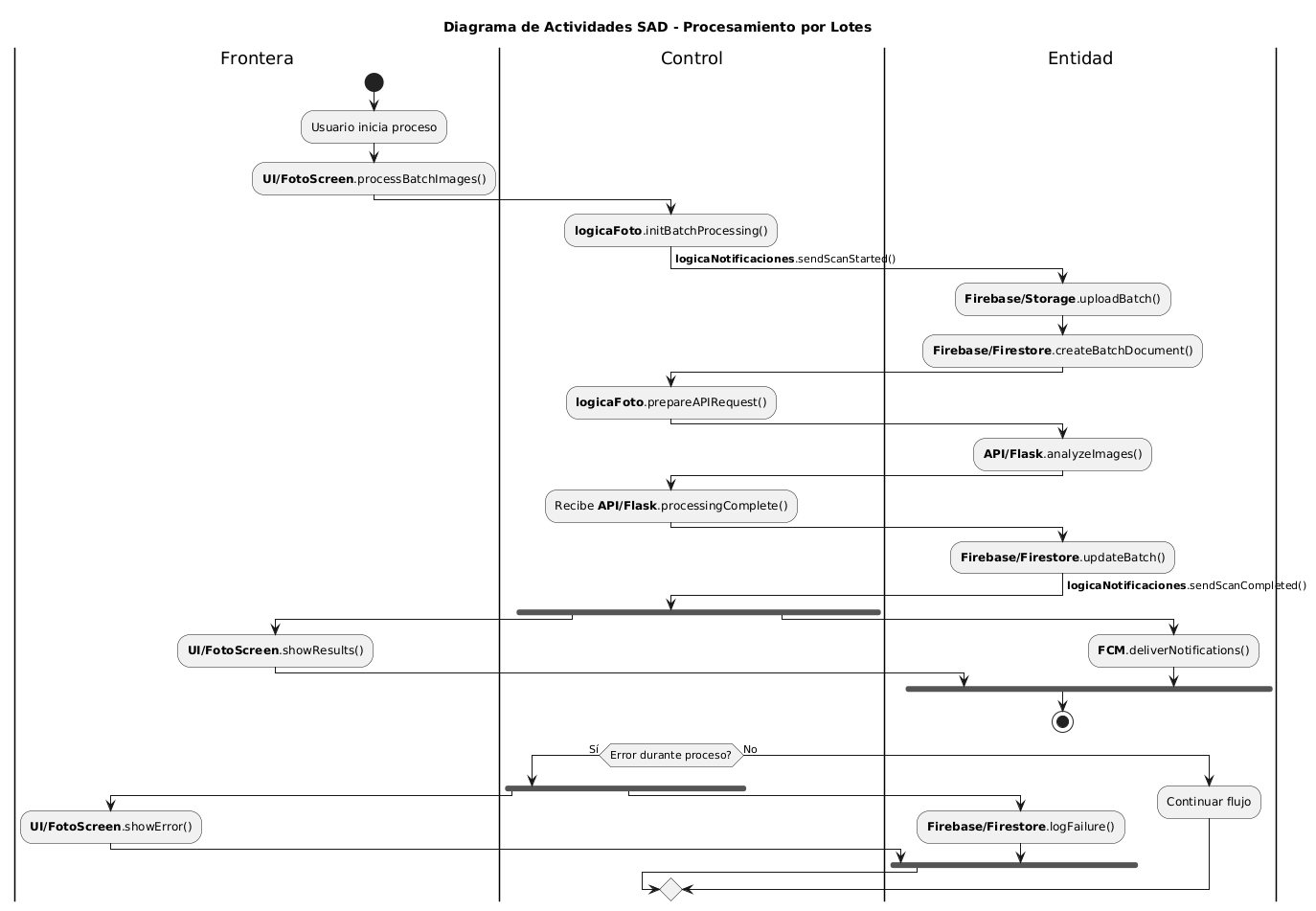
*Diagrama de Actividades con objetos 03 - Almacenar Historial de Escaneos:*

**

*Figura N°23: Diagrama de Actividades con objetos 03 - Almacenar Historial de Escaneos*

*Nota: Este diagrama detalla cómo el sistema guarda automáticamente cada análisis (imagen, diagnóstico, fecha y enfermedad) en el historial cuando el usuario está registrado.*

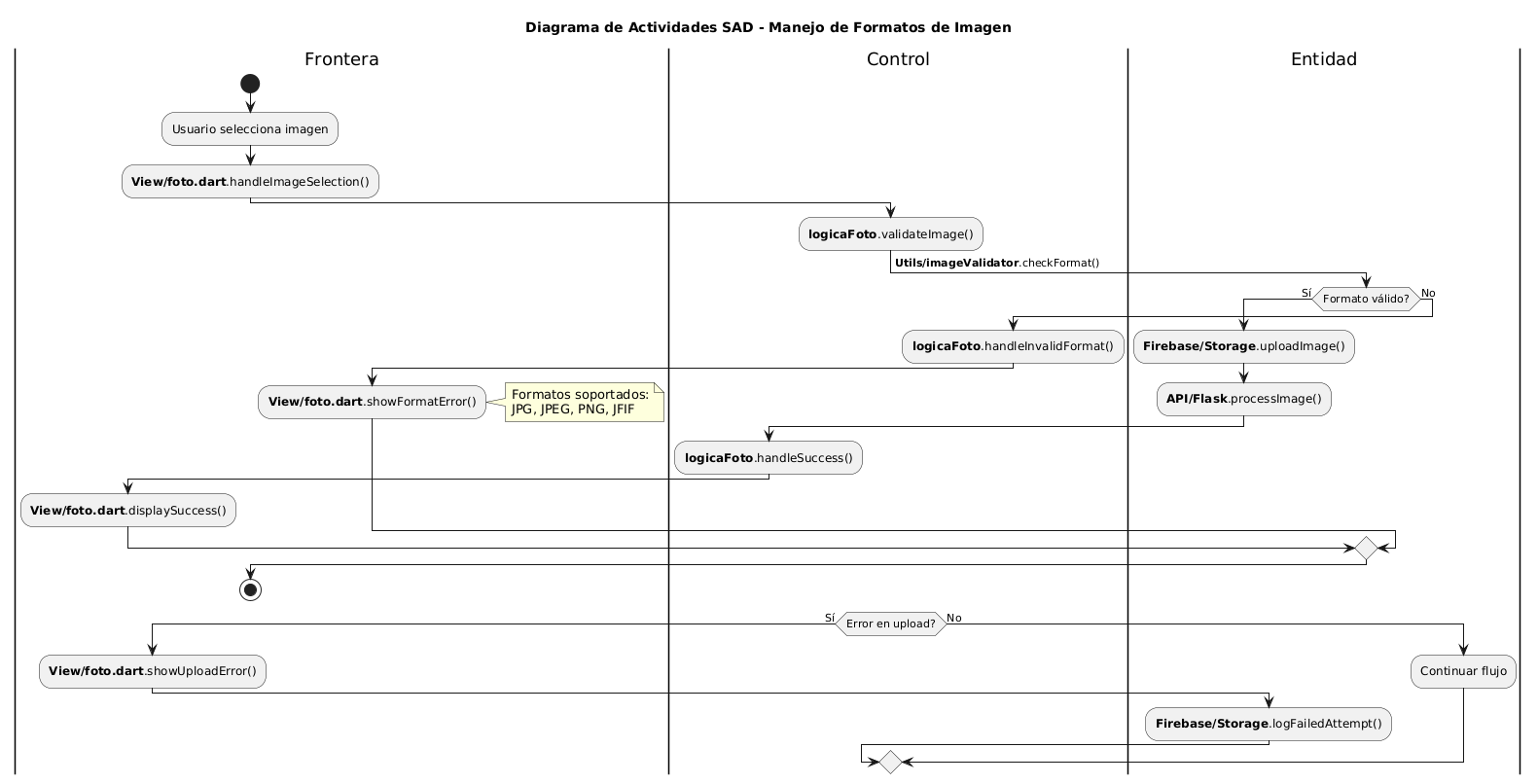
*Diagrama de Actividades con objetos 04 - Analizar Imágenes por Lote:*

**

*Figura N°24: Diagrama de Actividades con objetos 04 - Analizar Imágenes por Lote*

*Nota: Aquí se ilustra el procesamiento eficiente de múltiples imágenes en un solo lote.*

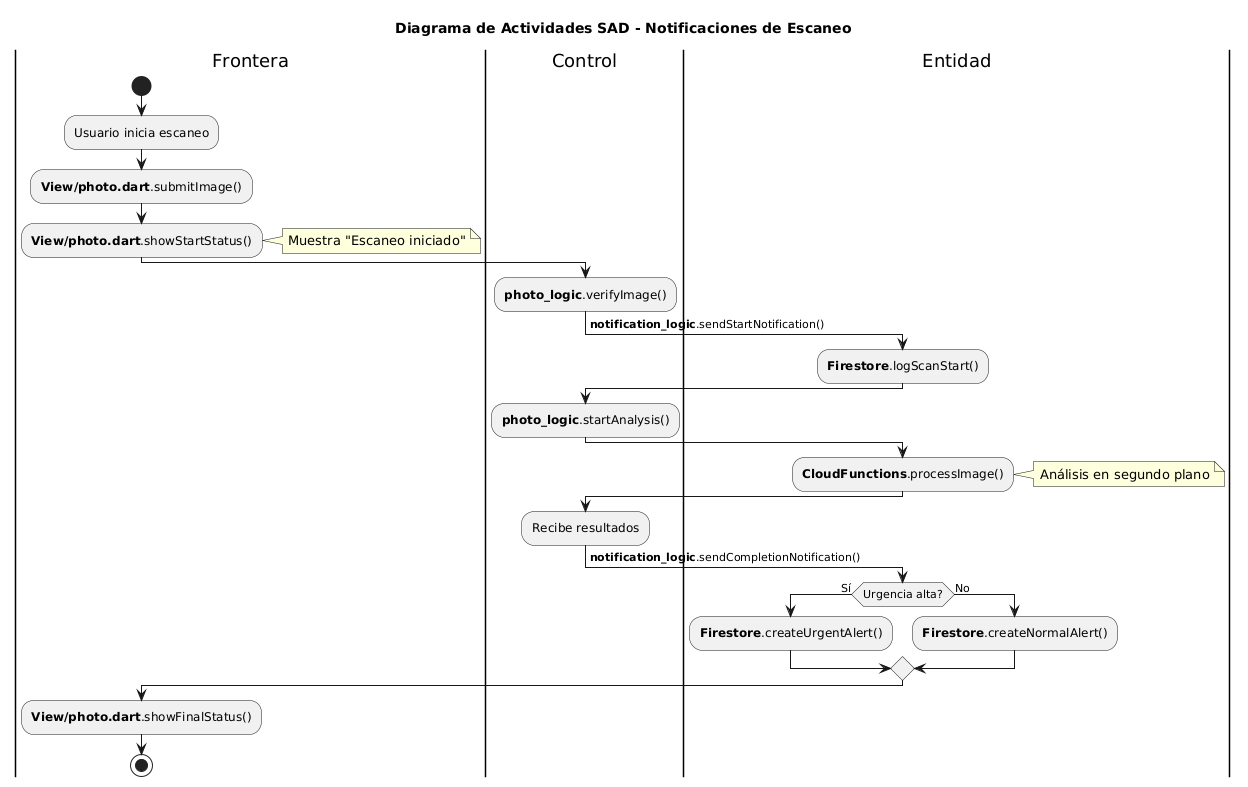
*Diagrama de Actividades con objetos 05 - Subir Imágenes en Diferentes Formatos:*

**

*Figura N°25: Diagrama de Actividades con objetos 05 - Subir Imágenes en Diferentes Formatos*

*Nota: El diagrama ilustra el proceso donde el usuario sube una imagen de su cultivo (JPG/PNG/GIF), el sistema verifica si el formato es válido, lo procesa si es compatible o muestra un mensaje de error claro si no lo es, asegurando una experiencia fluida al cargar imágenes sin problemas técnicos.*

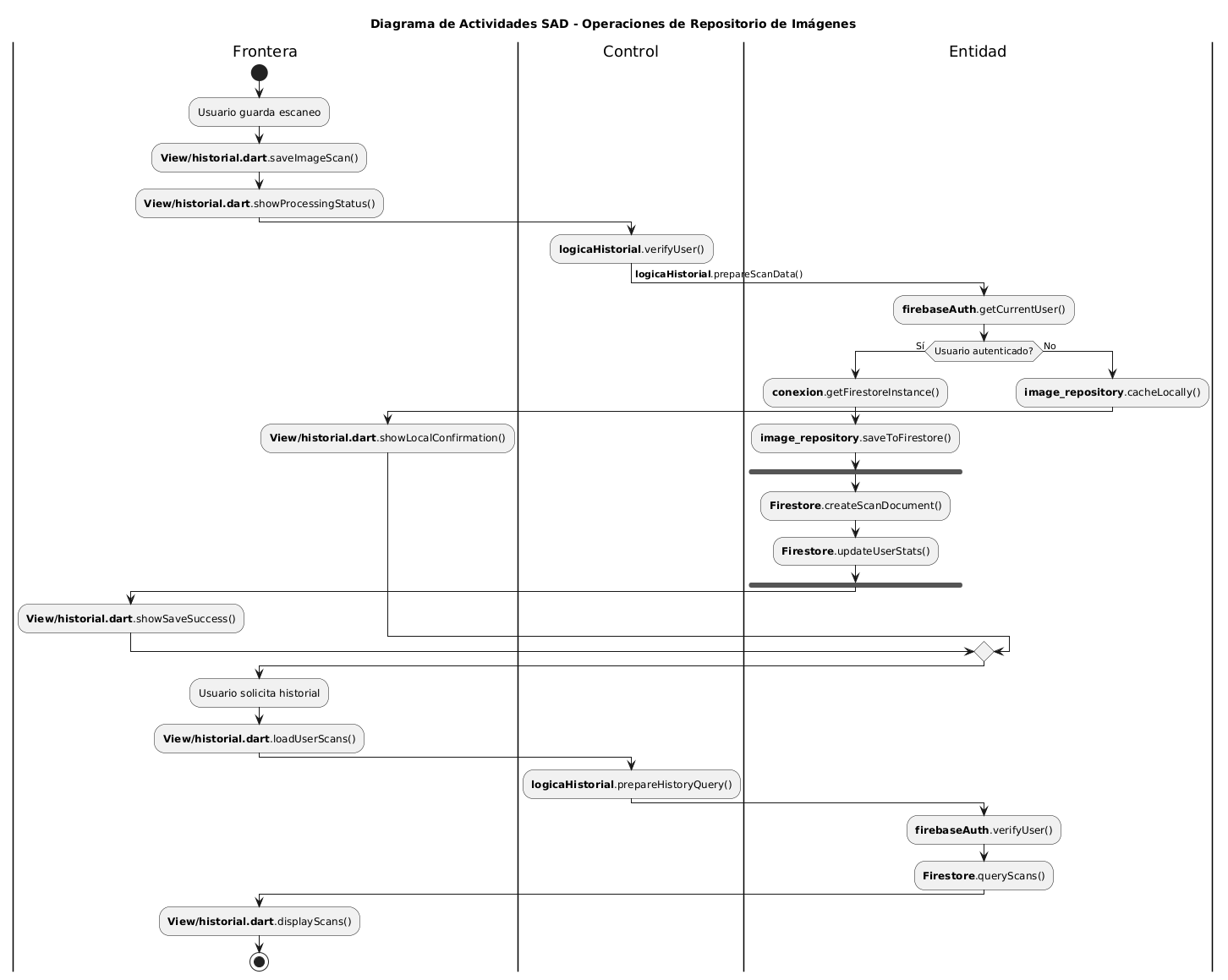
*Diagrama de Actividades con objetos 06 - Recibir Notificación de Análisis:*

**

*Figura N°26: Diagrama de Actividades con objetos 06 - Recibir Notificaciones de Análisis*

*Nota: Este flujo muestra cómo el sistema notifica al usuario en tiempo real sobre los resultados de sus análisis. Si se detecta una enfermedad, se envía una alerta urgente.*

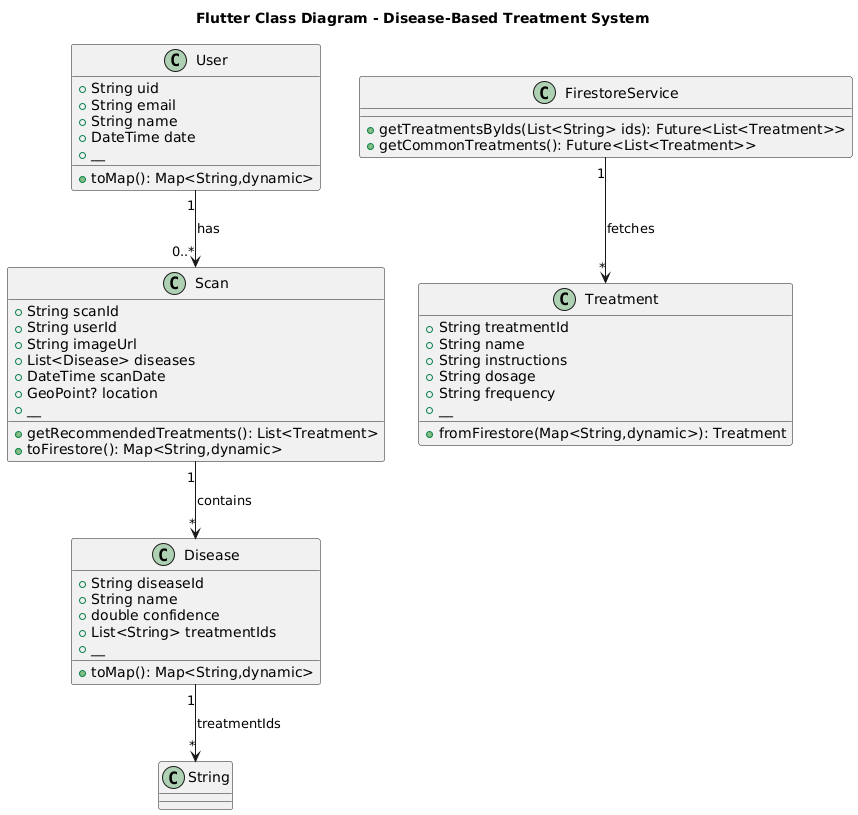
*Diagrama de Actividades con objetos 07 - Almacenar Datos en la Nube:*

**

*Figura N°27: Diagrama de Actividades con objetos 07 - Almacenar Datos en la Nube*

*Nota: Este flujo describe cómo la app guarda los escaneos e información del usuario en una base de datos en la nube, permitiendo acceder al historial desde cualquier dispositivo de forma instantánea, lo que facilita el monitoreo continuo de los cultivos sin depender de un solo equipo.*

* + 1. Diagrama de Clases



*Figura N°28: Diagrama de clases*

* + 1. Diagrama de Base de datos (relacional o no relacional)

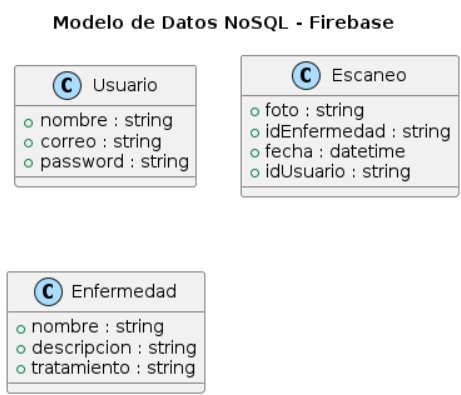
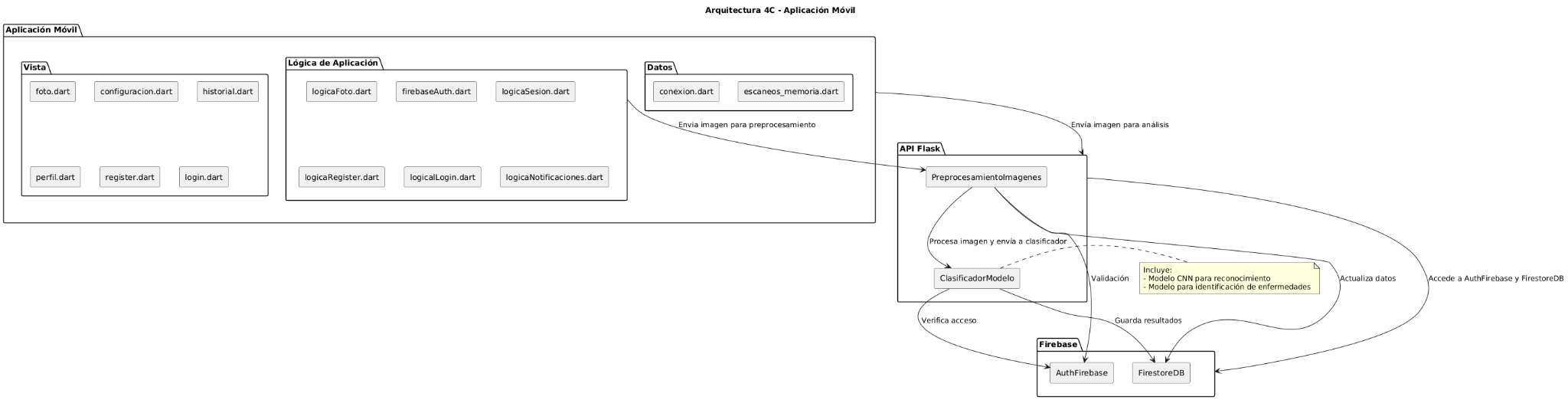


Figura N°29: Diagrama de Entidad-Relación

* 1. Vista de Implementación (vista de desarrollo)
     1. Diagrama de arquitectura software (paquetes)



*Figura N°30: Diagrama de arquitectura de software*

* + 1. Diagrama de arquitectura del sistema (Diagrama de componentes)

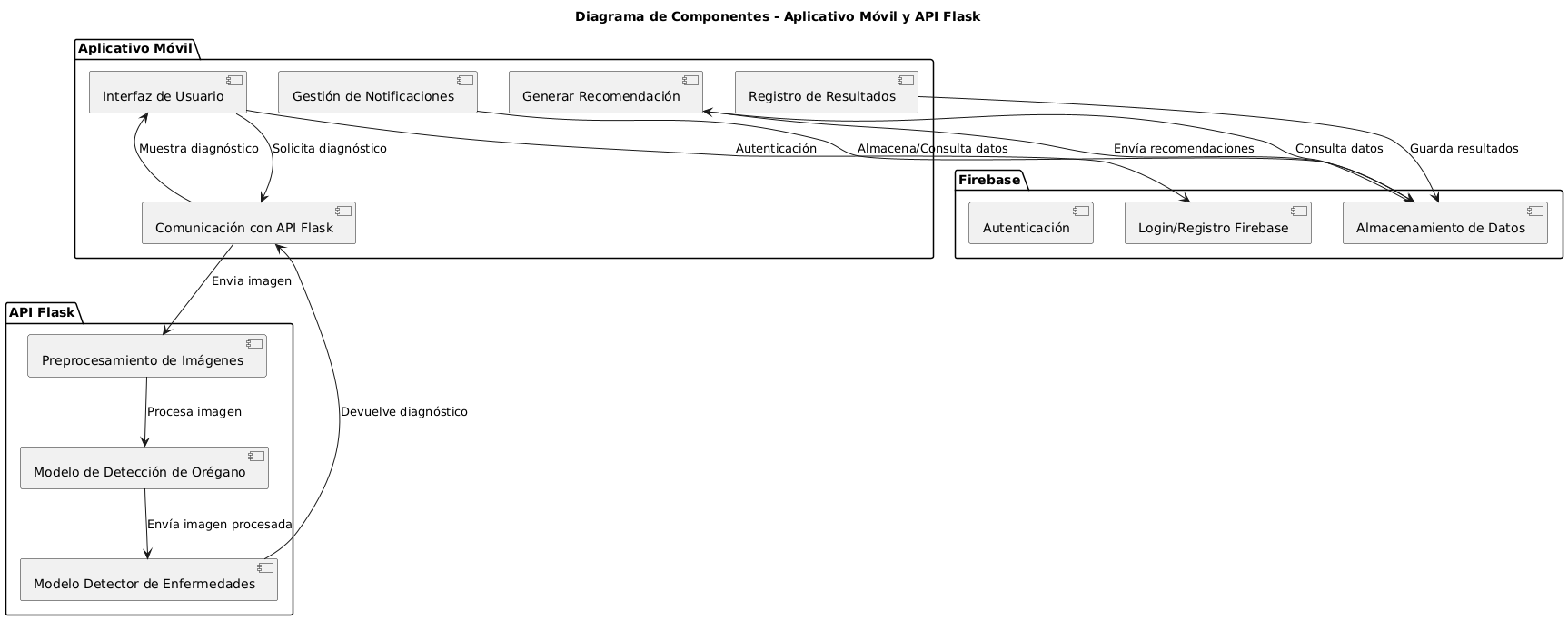
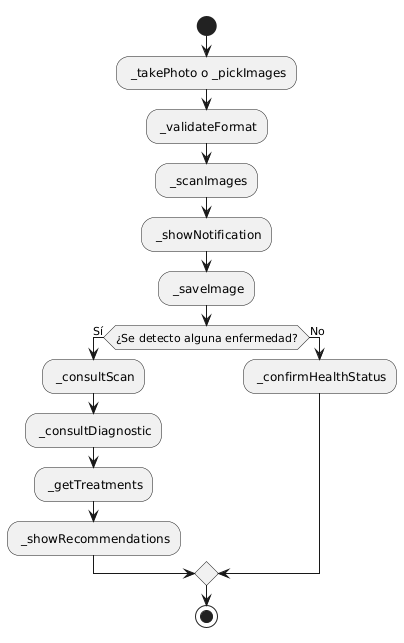


Figura N°31: Diagrama de componentes

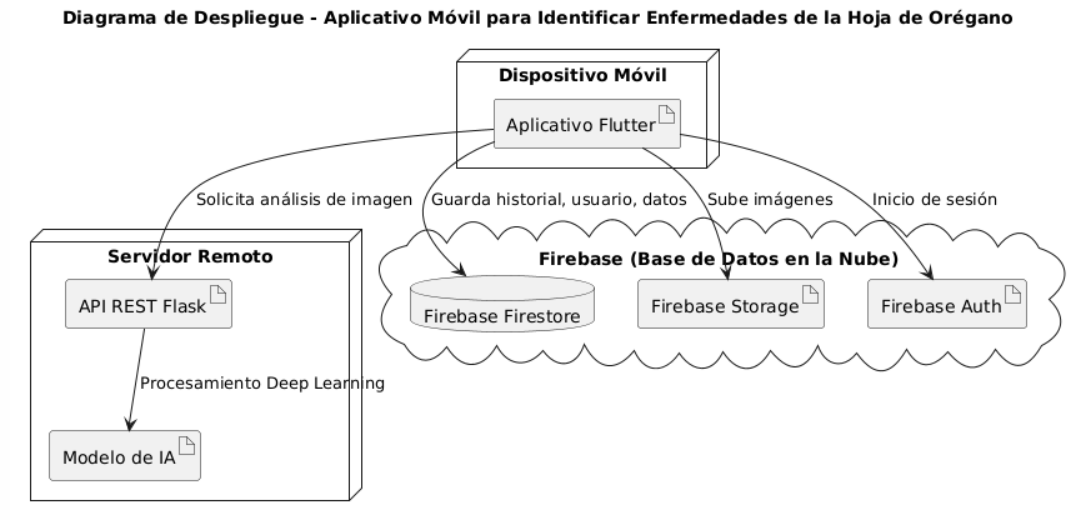
* 1. Vista de procesos
     1. Diagrama de Procesos del sistema (diagrama de actividad)



*Figura N°: Diagrama de Procesos del Sistema*

*Nota: El proceso inicia con la captura de imágenes mediante \_takePhoto (tomar foto) o \_pickImages (seleccionar imagen), seguido de \_validateFormat (validar formato). Luego, el aplicativo móvil ejecuta el escaneo \_scanImages, muestra una notificación de progreso \_showNotification y guarda los datos \_saveImage. Si se detectan enfermedades ¿Enfermedad? = (Sí), se activa el diagnóstico completo \_consultScan y \_consultDiagnostic, se generan tratamientos \_getTreatments y se muestran recomendaciones \_showRecommendations. En caso contrario (No), se confirma el estado saludable \_confirmHealthStatus.*

* 1. Vista de Despliegue (vista física)
     1. Diagrama de despliegue

**

*Figura N°32: Diagrama de despliegue*

1. **ATRIBUTOS DE CALIDAD DEL SOFTWARE**

**Escenario de Funcionalidad**

**Objetivo:**

Asegurar que el sistema cumpla con todas las funcionalidades esenciales para la detección de enfermedades en orégano, desde la captura de imágenes hasta la generación de recomendaciones, garantizando un flujo completo y eficiente.

Descripción Detallada:

La aplicación permite capturar imágenes de hojas de orégano mediante la cámara del dispositivo o seleccionarlas de la galería, validando automáticamente su calidad (enfoque, iluminación y formato compatible). Estas imágenes son procesadas por el modelo oreganoAI, que identifica enfermedades comunes (como la Roya) con un alto porcentaje de confianza y genera recomendaciones de tratamiento personalizadas, incluyendo dosis, frecuencia y productos específicos. Todos los resultados se almacenan en Firestore, permitiendo su acceso rápido, y se complementan con notificaciones en tiempo real para alertar sobre escaneos realizados. Además, el sistema maneja errores de conexión mediante caché local y reintentos automáticos, asegurando una experiencia fluida en cualquier condición.

**Métricas de Éxito:**

* Precisión del diagnóstico: ≥90% de aciertos.
* Tiempo de procesamiento: ≤15 segundos por imagen.
* Entrega de notificaciones: ≤5 segundos tras el diagnóstico.
* Tasa de capturas válidas: ≥75% en primer intento.

**Escenario de Usabilidad**

**Objetivo:**

Garantizar que la aplicación sea intuitiva y accesible para agricultores con diversos niveles de habilidad técnica, desde usuarios principiantes hasta avanzados, enfocándose en una experiencia fluida y satisfactoria en todas las interacciones.

**Descripción Detallada:**

El usuario inicia sesión de manera sencilla mediante correo electrónico y contraseña o autenticación rápida con Google. La interfaz muestra un botón prominente para capturar imágenes, acompañado de una guía visual que ayuda a encuadrar correctamente las hojas de orégano. Si la imagen está borrosa o mal iluminada, el sistema proporciona retroalimentación inmediata con instrucciones claras para mejorar la captura.

Una vez procesada la imagen, los resultados se presentan de forma visual con colores e iconos intuitivos: rojo para enfermedades críticas, amarillo para advertencias y verde para plantas saludables. Cada diagnóstico incluye recomendaciones de tratamiento detalladas, con pasos simples, dosis exactas y frecuencia de aplicación. El historial de escaneos se organiza de manera cronológica y permite filtrar por tipo de enfermedad o fecha, con una interfaz de desplazamiento suave y rápida.

**Métricas de Éxito:**

* Tiempo promedio para completar un escaneo: menos de 1 minuto.
* Reducción de errores en la captura de imágenes gracias a la validación automática.
* Satisfacción del usuario medida mediante encuestas (meta: ≥ 90% de usuarios satisfechos).

**Escenario de confiabilidad**

**Objetivo:**

Asegurar que la aplicación funcione de manera consistente y sin errores en diversas condiciones, incluyendo falta de conexión a internet, alta demanda del sistema o fallos en componentes externos.

**Descripción Detallada:**

En situaciones sin conexión a internet, la aplicación permite capturar y guardar escaneos localmente, sincronizándolos automáticamente cuando se restablece la conexión. Si el modelo de IA no puede procesar una imagen debido a baja calidad o formato incompatible, el sistema muestra un mensaje claro y sugiere acciones correctivas, como tomar una nueva foto o ajustar la iluminación.

Para manejar fallos en servicios externos como Firebase, la aplicación implementa un sistema de reintentos automáticos (hasta 3 veces) antes de notificar al usuario. Además, se incluyen mecanismos de recuperación ante errores críticos, como reinicios suaves de módulos afectados sin necesidad de cerrar la aplicación por completo.

**Métricas de Éxito:**

* Tasa de éxito de escaneos: superior al 95%.
* Tiempo de recuperación ante fallos: menos de 5 segundos.
* Disponibilidad del sistema: ≥ 95.0% (monitoreada mediante Firebase Crashlytics).

**Escenario de rendimiento**

**Objetivo:**

Optimizar la velocidad y eficiencia del sistema para garantizar tiempos de respuesta rápidos y un uso eficiente de los recursos del dispositivo, incluso en hardware de gama baja.

**Descripción Detallada:**

El procesamiento de imágenes individuales debe completarse en un máximo de 10 segundos, mientras que los lotes de hasta 5 imágenes no deben exceder 1 minuto. La carga del historial de escaneos, que puede incluir cientos de registros, se maneja mediante paginación, asegurando que los primeros 100 registros se muestren en menos de 3 segundos.

El consumo de memoria se mantiene por debajo de los 4096 MB en dispositivos básicos, y se evita el uso excesivo de la CPU durante operaciones prolongadas. Las imágenes se comprimen antes de subirlas a Firebase Storage para reducir el ancho de banda y los tiempos de transferencia.

**Métricas de Éxito:**

* Tiempo de análisis por imagen: ≤ 10 segundos.
* Tiempo de carga del historial: ≤ 10 segundos para 100 registros.
* Consumo de memoria: < 4096 MB en dispositivos de gama baja.

**Escenario de mantenibilidad**

**Objetivo:**

Facilitar la actualización, corrección de errores y escalabilidad del código, permitiendo que el equipo de desarrollo implementa nuevas características con un esfuerzo mínimo.

**Descripción Detallada:**

La arquitectura del sistema sigue principios de modularidad y separación de preocupaciones. La lógica de negocio (como el procesamiento de imágenes) se separa claramente de la interfaz de usuario y los servicios externos (Firebase, API Flask). Cada módulo incluye documentación detallada, tanto en el código como en un README.md, explicando su funcionalidad y dependencias.

Las pruebas automatizadas cubren las funciones críticas, como la validación de imágenes, el procesamiento de lotes y la generación de recomendaciones. Estas pruebas se ejecutan en un pipeline de CI/CD (Integración Continua/Despliegue Continuo) para detectar errores antes de llegar a producción.

**Métricas de Éxito:**

* Cobertura de pruebas: ≥ 80% del código crítico.
* Tiempo promedio para corregir errores: < 24 horas para problemas de prioridad alta.
* Frecuencia de actualizaciones: despliegues semanales sin interrupciones.

**Otros Escenarios**

**Seguridad**

**Objetivo:**

Proteger los datos sensibles de los usuarios y garantizar que solo personas autorizadas puedan acceder a la información de cultivos y diagnósticos.

**Descripción Detallada:**

La autenticación se gestiona mediante Firebase Auth, con reglas estrictas en Firestore que restringen el acceso a los datos. Por ejemplo, un usuario solo puede leer o escribir sus propios escaneos. Las contraseñas se almacenan de forma segura usando hash, y todas las comunicaciones con la API Flask se realizan mediante HTTPS para evitar interceptaciones.

Las imágenes subidas a Firebase Storage se protegen con permisos específicos por usuario, y los datos sensibles, como ubicaciones GPS, se anonimizan antes de almacenarse. Además, se implementa un sistema de auditoría para registrar accesos sospechosos o intentos de violación de seguridad.

**Métricas de Éxito:**

* Cero vulnerabilidades críticas reportadas en pruebas de penetración.
* Cumplimiento de normativas de privacidad.