**[ColeccionAPP]**

**(DAS) Documento Arquitectura Sistema**

**Versión 1.1**

**Identificación de Documento**

|  |  |
| --- | --- |
| **Identificación** |  |
| **Proyecto** | ColeccionAPP |
| **Versión** | 1.1 |

|  |  |
| --- | --- |
| **Documento mantenido por** | Diego Quiroz |
| **Fecha de última revisión** | 23/10/2025 |
| **Fecha de próxima revisión** |  |

|  |  |
| --- | --- |
| **Documento aprobado por** | Diego Quiroz |
| **Fecha de última aprobación** | 19/11/2025 |

**Historia de Revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 23/10/2025 | 1.0 | Se comienza documento | Diego Quiroz |
| 18/11/2025 | 1.1 | Se completa documento | Noelia Peña |
|  |  |  |  |
|  |  |  |  |

**Tabla de Contenidos**

[**1**](#_heading=h.q4mxcjq48nyf) **Introducción 3**

[1.1](#_heading=h.y4l3oqz5ffos) Contexto del Problema 3

[1.2](#_heading=h.qw7jeif81zq) Propósito 3

[1.3](#_heading=h.fudt9fc1urqy) Ámbito 3

[1.4](#_heading=h.i8q9hh6kw251) Definiciones, acrónimos y abreviaciones 3

[1.5](#_heading=h.x4ygtxgvt0ll) Referencias 3

[1.6](#_heading=h.927ki2ww3xrs) Resumen ejecutivo 3

[1.7](#_heading=h.3foxvwe17p2d) Representación 4

[**2**](#_heading=h.u5pdn1bdf3j5) **Metas y Restricciones de la Arquitectura 5**

[2.1](#_heading=h.3wt0ksmpqegv) Metas de la arquitectura 5

[2.2](#_heading=h.wwlk4s5bvjma) Restricciones de la Arquitectura 5

[2.3](#_heading=h.82yjmnthf13l) Otros antecedentes y consideraciones 5

**3** **Vista de Escenarios 6**

[3.1](#_heading=h.vzrh9u3gbkzr) Modelo de Casos de Uso 6

[3.2](#_heading=h.6uqcsp6o1nl) Casos de Usos Extendidos 6

[3.3](#_heading=h.2le2xca0vm2q) Especificación de los Escenarios de Calidad Relevantes 7

[**4**](#_heading=h.91cds59f4d5j) **Vista de Procesos 8**

[**5**](#_heading=h.xkzwf4u4b83w) **Vista Lógica 9**

[5.1](#_heading=h.r81wnxcueq2i) Parte Estructural ( Diagrama de Clases y Diagrama Relacional) 9

[*5.1.1*](#_heading=h.omsda9vjqi26) *Descripción de Clases 9*

[*5.1.2*](#_heading=h.yj8st7mwnn1q) *Descripción de Tablas 10*

[5.2](#_heading=h.b9rr7c97u7e2) Parte Dinámica (Diagrama de Secuencias) 11

[**6**](#_heading=h.xk0gbxtb8u6f) **Vista de Desarrollo o Despliegue 12**

[**7**](#_heading=h.25tj2iakg4wq) **Vista Fisica 13**

[**8**](#_heading=h.ys5wxme2nnnh) **Decisiones de Diseño y Selección de Alternativas 14**

[**9**](#_heading=h.ldff5jbkaker) **Análisis de Reutilización 15**

1. **Introducción**

Contexto del Problema

En los últimos años, la denominada *Cultura Geek* ha adquirido una presencia significativa en la vida cotidiana, impulsada por fenómenos culturales globales como el Universo Cinematográfico de Marvel, el auge continuo de franquicias como Pokémon y la masificación del coleccionismo como hobby.

Este incremento de coleccionistas, en especial adultos jóvenes con poder adquisitivo, ha generado la necesidad de administrar de manera eficiente grandes cantidades de ítems adquiridos, deseados o en seguimiento. Sin embargo, la gestión de colecciones suele realizarse manualmente mediante planillas o notas personales, lo que produce:

Pérdida de control sobre los artículos adquiridos.

Dificultad para recordar qué productos están disponibles en el mercado.

Riesgo de compras duplicadas.

Ausencia de una base unificada de productos.

Como consecuencia, los coleccionistas experimentan problemas económicos, organizacionales y de seguimiento.  
ColeccionAPP surge como respuesta directa a esta problemática, entregando una solución tecnológica moderna, automatizada y multiplataforma.

* 1. **Propósito**

El propósito de este Documento de Arquitectura del Sistema (DAS) es definir formalmente la estructura técnica, las decisiones de diseño y los componentes que conformarán *ColeccionAPP*.

El DAS establece:

* La arquitectura lógica, física y de despliegue.
* Las vistas requeridas por el modelo 4+1.
* Las decisiones tecnológicas fundamentales.
* La relación entre los componentes internos, externos y servicios cloud utilizados.
* La correspondencia con los requerimientos funcionales y no funcionales definidos en el ERS.

Este documento guía al equipo de desarrollo y asegura que la solución propuesta soporte todos los requisitos del cliente, incluyendo el uso de **IA para reconocimiento de imágenes mediante la API de Roboflow**, conforme al ERS.

* 1. **Ámbito**

El alcance del sistema abarca:

* Aplicación móvil multiplataforma desarrollada con Flutter (Android, iOS).
* Registro y autenticación de usuarios mediante Firebase Authentication (con compatibilidad Google Sign-In).
* Gestión completa de colecciones personales.
* Registro manual y automático (IA) de productos.
* Módulo de búsqueda y filtrado avanzado.
* Repositorio estructurado de productos disponibles en el mercado.
* Clasificación automática basada en imágenes usando Roboflow API.

El sistema se limita inicialmente a ciertos tipos de coleccionables definidos en el ERS (dos expansiones Digimon TCG y figuras Storm Collectibles 2024–2025), con posibilidad de ampliación futura.

* 1. **Definiciones, acrónimos y abreviaciones**

|  |  |
| --- | --- |
| **ACRONIMO** | **DESCRIPCION** |
| *MVC* | Modelo-Vista-Controlador |
| *CRUD* | Create, Read, Update, Delete |
| *IA* | Inteligencia Artificial |
| *FIREBASE* | Plataforma cloud de Google utilizada para BD, autenticación y almacenamiento |
| *FIRESTORE* | Base de datos NoSQL provista por Firebase |
| *ROBOFLOW* | Plataforma de entrenamiento y servición de modelos IA |
| *API* | Application Programming Interface |
| *SDK* | Software Development Kit |
|  |  |

* 1. **Referencias**

A continuación, se listan las referencias a otros documentos:

* ERS – Especificación de Requisitos de Software del proyecto ColeccionAPP.

Informe ERS

* Planilla de requerimientos funcionales y no funcionales.
* Planificación del proyecto (Gantt y EDT).
* Documento de casos de uso extendido.
* Mockups y prototipo de interfaces.

* 1. **Resumen ejecutivo**

ColeccionAPP es una aplicación móvil multiplataforma orientada a coleccionistas, cuyo objetivo es facilitar el registro y administración de sus colecciones mediante:

* Un sistema de gestión intuitivo.
* Un repositorio actualizado de productos.
* Un sistema de clasificación automático basado en imágenes.
* Servicios en la nube (Firebase) que permiten escalabilidad, seguridad y persistencia.
* Arquitectura modular consistente con el tiempo de desarrollo y recursos disponibles.

El sistema integra un modelo de IA (IA externa mediante Roboflow API) que reconoce productos a partir de fotografías, permitiendo registrar artículos de forma automática. Esto diferencia a ColeccionAPP de hojas de cálculo u otros métodos manuales utilizados actualmente.

* 1. **Representación**

La arquitectura del sistema ColleccionAPP está representada siguiendo el enfoque de del framework 4+1 y las recomendaciones del proceso unificado. Las vistas incluidas en esta versión del documento son:

* **Vista de Escenarios**: Describe los casos de uso más significativos, presenta los actores y una descripción de sus casos de uso asociados. De igual forma describe los escenarios de calidad más relevantes para la arquitectura.
* **Vista de Procesos**: Describe los procesos involucrados para darle sentido a la ejecución del sistema, así como sus relaciones de comunicación y sincronización.
* **Vista Lógica**: Describe la arquitectura del sistema presentando varios niveles de refinamiento. Indica los módulos lógicos principales, sus responsabilidades y dependencias.
* **Vista de Desarrollo o Despliegue**: Describe los componentes de deployment construidos y sus dependencias.
* **Vista Fisica**: Describe restricciones tecnológicas, normativas, estándares, etc., los cuales influyen sobre las decisiones arquitectónicas, del producto y del proceso de desarrollo.

1. **Metas y Restricciones de la Arquitectura**
   1. **Metas de la arquitectura**

Las metas arquitectónicas del sistema *ColeccionAPP* se definen considerando los requerimientos funcionales, no funcionales y los escenarios de calidad descritos en el ERS. Estas metas guían las decisiones de diseño para garantizar un sistema confiable, escalable y mantenible.

**Desempeño**

* El sistema debe responder en tiempo real durante operaciones críticas como autenticación, búsqueda de productos y registro automático por IA.
* Cumplir objetivos de latencia del ERS:
  + Inicio de aplicación ≤ 5 s
  + Búsqueda de productos ≤ 3 s
  + Clasificación IA ≤ 7 s

**Escalabilidad**

* Arquitectura basada en servicios de Firebase y API externa de Roboflow permite escalar en usuarios y volumen de datos sin reestructuras mayores.
* Soporta integración futura de módulos sociales, web-app y dashboards administrativos.

**Seguridad**

* Autenticación confiable mediante Firebase Authentication.
* Encriptación TLS 1.2+ en tránsito y AES-256 en reposo.
* Control de acceso por roles (usuario / administrador).
* Manejo seguro de datos sensibles según lineamientos del ERS y Google Cloud.

**Mantenibilidad**

* Arquitectura modular por paquetes (Usuario, Colecciones, Productos, Repositorio, Clasificación IA).
* Separación clara de responsabilidades siguiendo un enfoque por componentes.
* Integración con CI/CD para facilitar pruebas, despliegues y refactorizaciones.

**Disponibilidad**

* Servicios Cloud que permiten disponibilidad ≥ 99% mensual según requisitos de calidad.
* Mínima dependencia en infraestructura propia.

**Usabilidad**

* Diseño intuitivo y adaptable a distintos dispositivos móviles.
* Cumple requisitos del ERS de personalización (tema, tamaño de letra, color).

**Restricciones de la Arquitectura**

Las restricciones son factores no negociables que condicionan la implementación del sistema. Estas restricciones provienen del cliente, del contexto de ejecución y del propio ERS.

Restricciones Tecnológicas

* Framework principal: Flutter + Dart (obligatorio).
* Backend: Firebase (Firestore, Storage, Authentication).
* IA: Se debe usar Roboflow API para la clasificación de imágenes.
* Sistema operativo: Android 24+ y iOS 13+.
* Sin licencias propietarias: el proyecto prioriza herramientas gratuitas o de código abierto.

Restricciones de Tiempo

* Tiempo total aproximado: 5 sprints (18–20 semanas).
* Desarrollo incremental:
* Base de la App sin IA
* Integración del modelo IA
* QA y certificación
* Release productivo

Restricciones Funcionales

* El sistema debe incluir registro automático con IA, sin depender de hardware externo.
* El repositorio inicial debe contener solo dos expansiones Digimon + figuras Storm Collectibles, según el ERS.

Restricciones de Infraestructura

* Firestore debe ser la base principal.
* Acceso a servicios Google y Roboflow a través de HTTPS.
* Sin backend propio tradicional: todo se maneja mediante servicios cloud o API externa.
  1. **Otros antecedentes y consideraciones**
* La precisión mínima del modelo IA debe ser ≥ 85% (según ERS).
* La app debe ser multiplataforma; sin embargo, en esta versión del DAS se omite el módulo Admin Web según tu instrucción.
* La arquitectura debe preparar el terreno para la expansión futura hacia integración social y dashboards.
* Toda comunicación entre la App, Firebase y Roboflow se realiza mediante API REST sobre HTTPS.
* Se considera soporte offline para lectura de colecciones gracias a capacidades de Firestore.

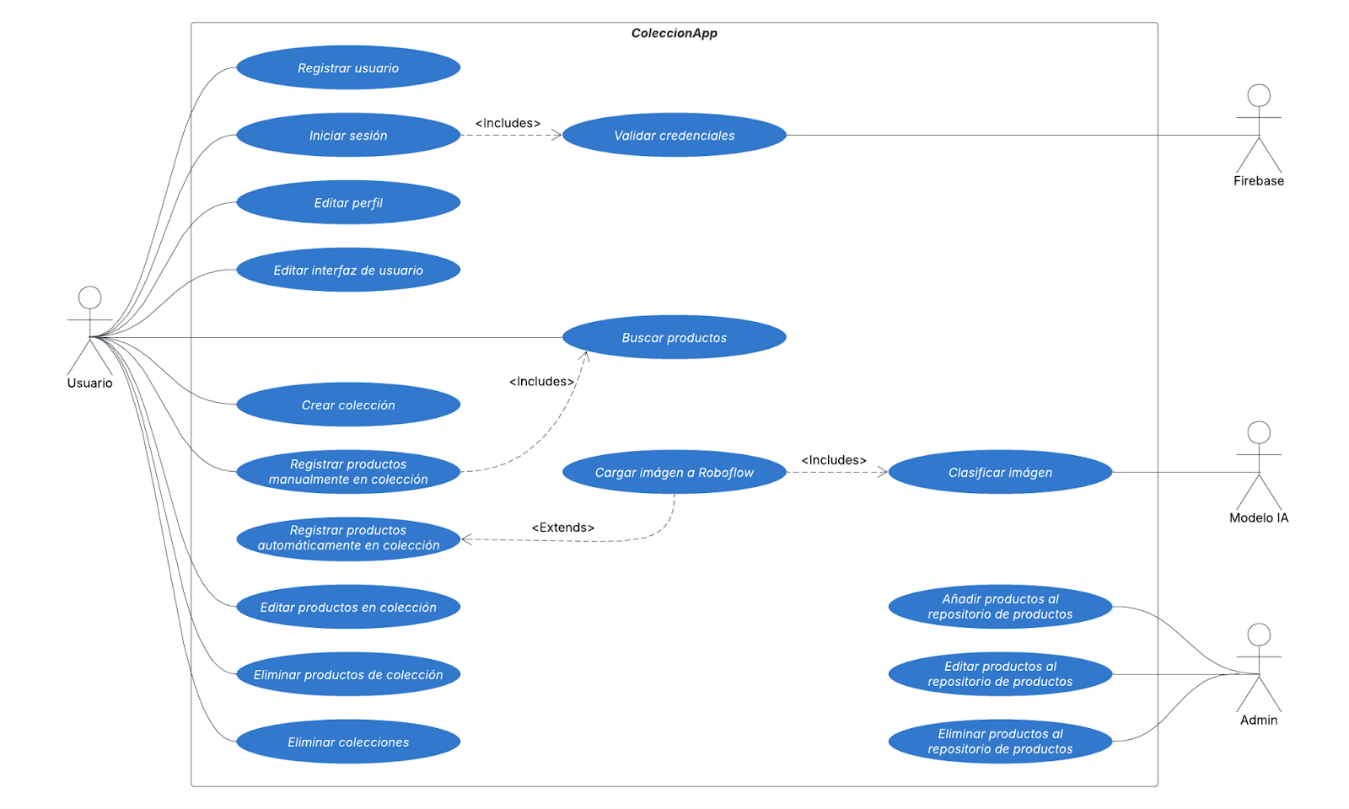
1. **Vista de Escenarios**

Esta sección describe en detalle el conjunto de escenarios funcionales y no funcionales que obtuvieron la mayor prioridad en el análisis. Para esto se presenta y describe el diagrama de casos de uso y los casos de uso prioritarios, así como los escenarios en que uno o más atributos de calidad se ven involucrados de manera significativa.

* 1. **Modelo de Casos de Uso**

Agregar el modelo de caso uso general del sistema

**Ilustración 1: Diagrama de Caso Uso General del Sistema**



* 1. **Casos de Usos Extendidos**

Los casos de uso considerados son los más relevantes para el desarrollo de la arquitectura. Se adjunta el documento o planilla caso uso.

**Adjuntar la planilla caso uso extendido**

A continuación, se listan los casos de uso relevantes, los cuales pueden ser encontrados con su especificación detallada en el documento “Casos de Uso Extendido”.

|  |  |  |  |
| --- | --- | --- | --- |
| **Código** | **Nombre** | **Actores** | **Prioridad** |
| CU-001 | Registrar usuario en la App sin cuenta de Google | Usuario Final | Alta |
| CU-002 | Iniciar sesión sin cuenta de Google | Todos los usuarios | Alta |
| CU-003 | Iniciar sesión con cuenta de Google | Usuario Final | Alta |
| CU-004 | Creación de colecciones/listas | Usuario Final | Alta |
| CU-005 | Administración de colecciones/listas – Editar | Usuario Final | Alta |
| CU-006 | Administración de colecciones/listas - Eliminar | Usuario Final | Alta |
| CU-007 | Registro manual de productos en colecciones/listas | Usuario Final | Alta |
| CU-008 | Registro automático de productos en colecciones/listas | Usuario Final | Alta |
| CU-009 | Administración de productos en colecciones/listas | Usuario Final | Alta |
| CU-010 | Administración de productos en colecciones/listas | Usuario Final | Alta |
| CU-011 | Búsqueda de productos en la aplicación | Usuario Final | Alta |
| CU-012 | Edición del perfil de usuario (Mi Perfil) | Usuario Final | Alta |
| CU-013 | Personalización de interfaz de usuario | Usuario Final | Media |

* 1. **Especificación de los Escenarios de Calidad Relevantes**

Los siguientes escenarios de calidad fueron priorizados tras el análisis con los stakeholders del proyecto Colección APP, considerando los atributos más críticos: rendimiento, seguridad y disponibilidad.

|  |  |  |
| --- | --- | --- |
| Identificador: EC-01 Reconocimiento automático con IA (Roboflow API) | | |
| Escenario(s): | | Clasificación de imagen al registrar un producto usando cámara. |
| Atributos de Calidad relevantes: | | Rendimiento / Eficiencia. |
| Componentes del Escenario | Estímulos: | El usuario toma una fotografía para registrar un producto automáticamente. |
| Fuente del estimulo | Usuario final (App móvil). |
| Ambiente: | Conectividad normal a internet (4G/5G/WiFi). Aplicación en estado activo. |
| Artefacto: | Módulo de Clasificación IA → Roboflow API. |
| Respuesta: | La imagen es enviada al servicio externo (Roboflow).  El modelo retorna predicción.  La App muestra al usuario el producto reconocido |
| Medida de Respuesta | Tiempo de clasificación **≤ 7 segundos** (ERS 3.3.1).  Precisión del modelo **≥ 85%** (ERS R.7) |

|  |  |  |
| --- | --- | --- |
| Identificador: EC-02 Seguridad: Inicio de sesión y protección de datos | | |
| Escenario(s): | | Garantizar que la autenticación no permita accesos no autorizados. |
| Atributos de Calidad relevantes: | | Seguridad |
| Componentes del Escenario | Estímulos: | Un usuario intenta autenticarse en el sistema (con Google o email/contraseña). |
| Fuente del estímulo | Usuario final. |
| Ambiente: | Aplicación ejecutándose en móvil, con canal HTTPS. |
| Artefacto: | Firebase Authentication. |
| Respuesta: | Se validan credenciales.  Se establece sesión segura.  Los datos personales se transmiten cifrados. |
| Medida de Respuesta | Autenticación protegida con **TLS 1.2+** (ERS 3.3.2).  Contraseñas gestionadas solo por Firebase (no almacenadas por la App).  Tasa de accesos no autorizados: **0%** |

|  |  |  |
| --- | --- | --- |
| Identificador: EC-03 Disponibilidad: Uso continuo de la App | | |
| Escenario(s): | | Los usuarios gestionan colecciones y productos sin interrupciones. |
| Atributos de Calidad relevantes: | | Disponibilidad |
| Componentes del Escenario | Estímulos: | El usuario intenta abrir la App y cargar “Mis Colecciones”. |
| Fuente del estímulo | Usuarios finales. |
| Ambiente: | Normal, carga desde Firestore. |
| Artefacto: | Firebase Firestore + módulos locales. |
| Respuesta: | Firestore devuelve datos de colecciones del usuario.  La app muestra correctamente sus listas. |
| Medida de Respuesta | Disponibilidad mensual de servicios ≥ **99%** (ERS 3.3.4).  Caída máxima permitida: **≤ 8 horas al mes**. |

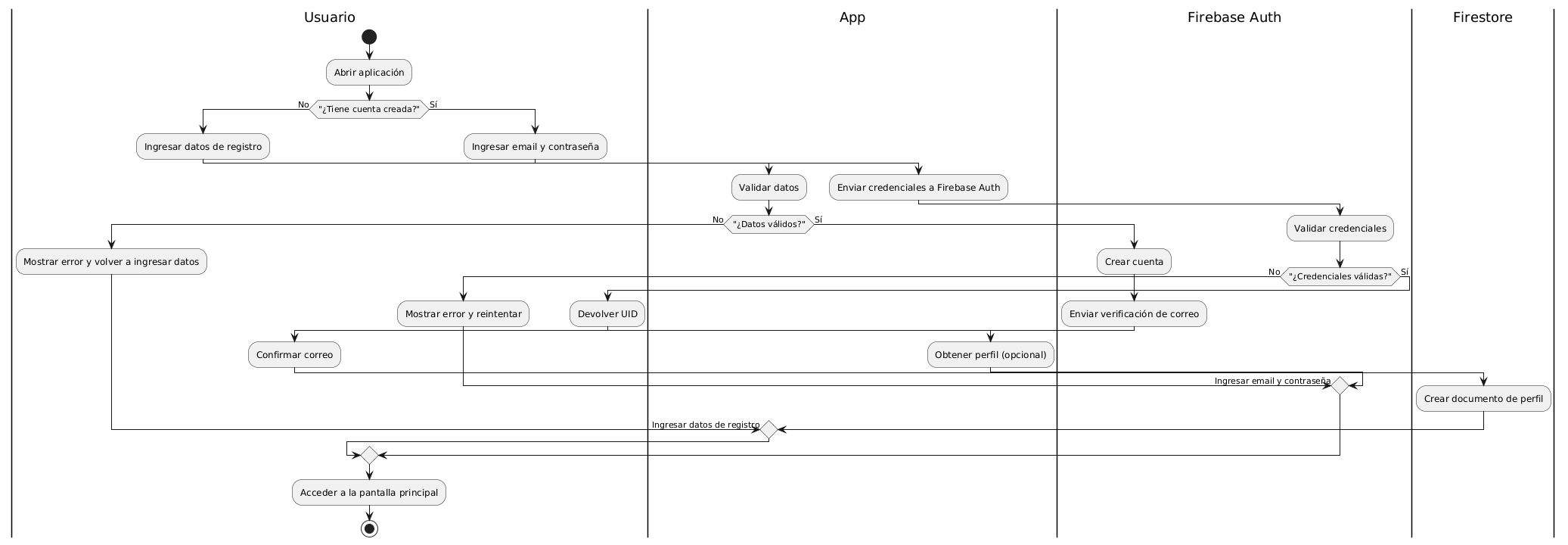
|  |  |  |
| --- | --- | --- |
| Identificador: EC-04 Usabilidad: Registro manual de un producto | | |
| Escenario(s): | | Registrar de manera manual objeto de colección Storm Arena. |
| Atributos de Calidad relevantes: | | Usabilidad |
| Componentes del Escenario | Estímulos: | Usuario quiere agregar manualmente un producto a una colección |
| Fuente del estímulo | Usuario Final |
| Ambiente: | Pantalla “Detalle de Colección” activa. |
| Artefacto: | UI de Búsqueda + Repositorio de Productos |
| Respuesta: | El usuario encuentra el producto por texto o filtros.  Se agrega a la colección facilmente. |
| Medida de Respuesta | Tiempo máximo para completar tarea **≤ 60 segundos** (ERS 3.3.3).  Tasa de éxito en tarea ≥ **95%**. |

|  |  |  |
| --- | --- | --- |
| Identificador: EC-05 Usabilidad en Funciones Principales | | |
| Escenario(s): | | Incremento del volumen de productos, usuarios y registros IA. |
| Atributos de Calidad relevantes: | | Usabilidad, Experiencia de usuario. |
| Componentes del Escenario | Estímulos: | Usuario ejecuta tareas clave del sistema. |
| Fuente del estímulo | Usuario Final |
| Ambiente: | Interfaz móvil Flutter. |
| Artefacto: | App ColeccionAPP + Firestore + módulos UI. |
| Respuesta: | nterfaz clara, carga rápida, navegación simple. |
| Medida de Respuesta | >95% tasa de éxito; tareas principales < 60 segundos. |

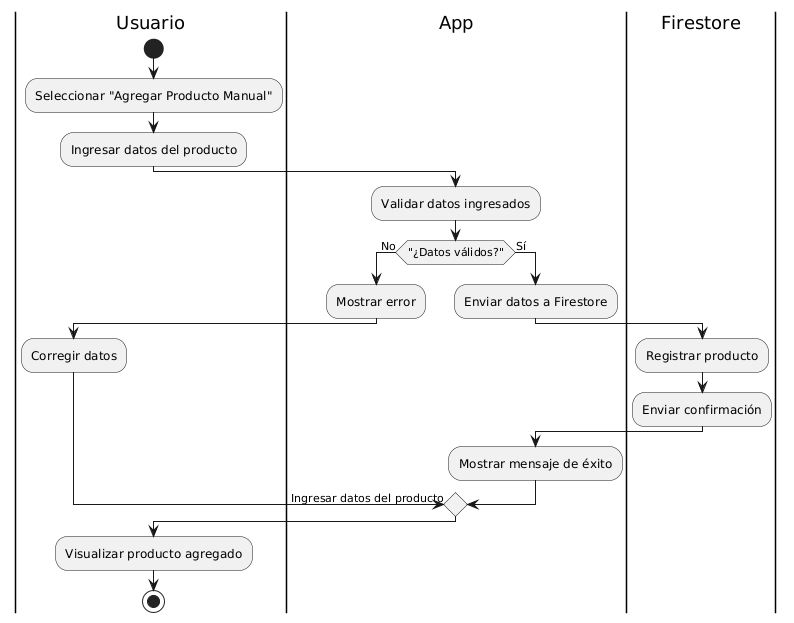
1. **Vista de Procesos**

**Ilustración 2: Diagramas de Actividades**

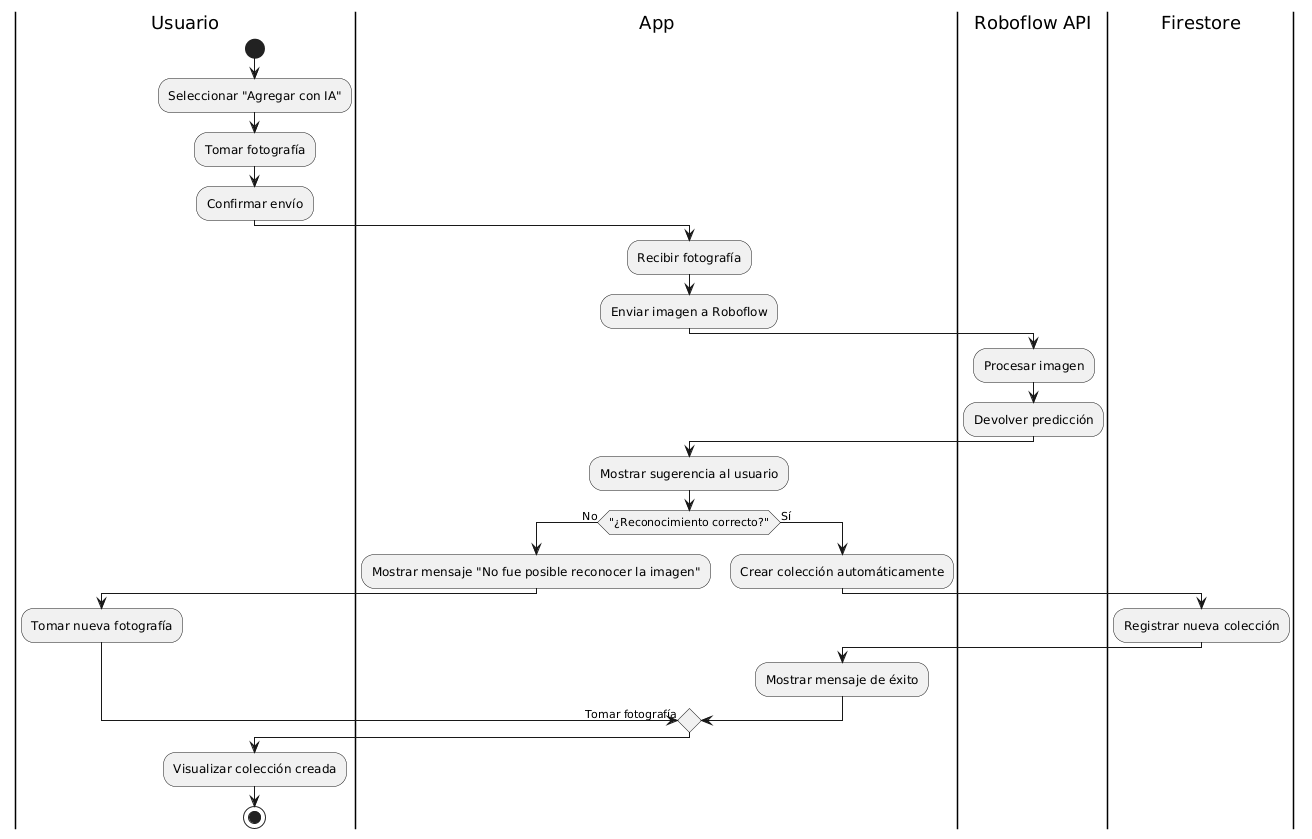
1. **Registro y autenticación de usuario.**



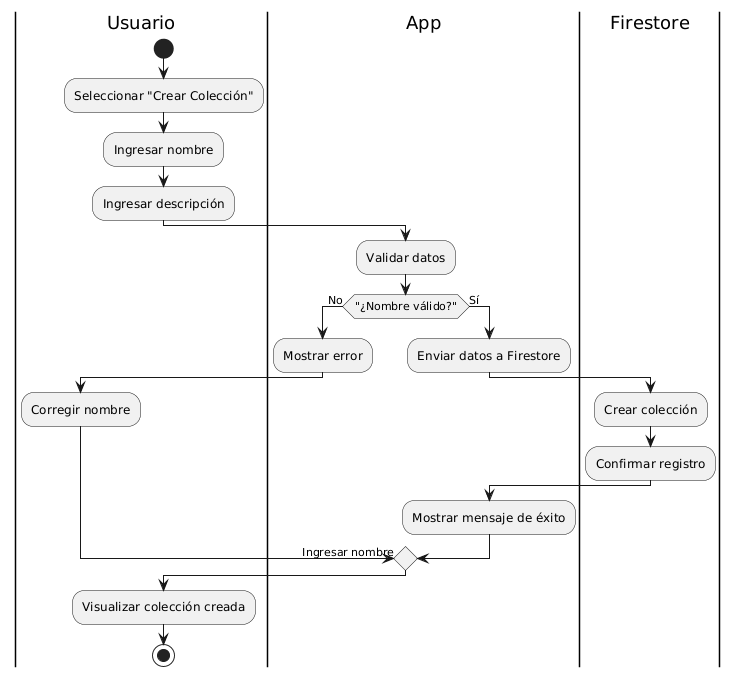
1. Registro manual de producto



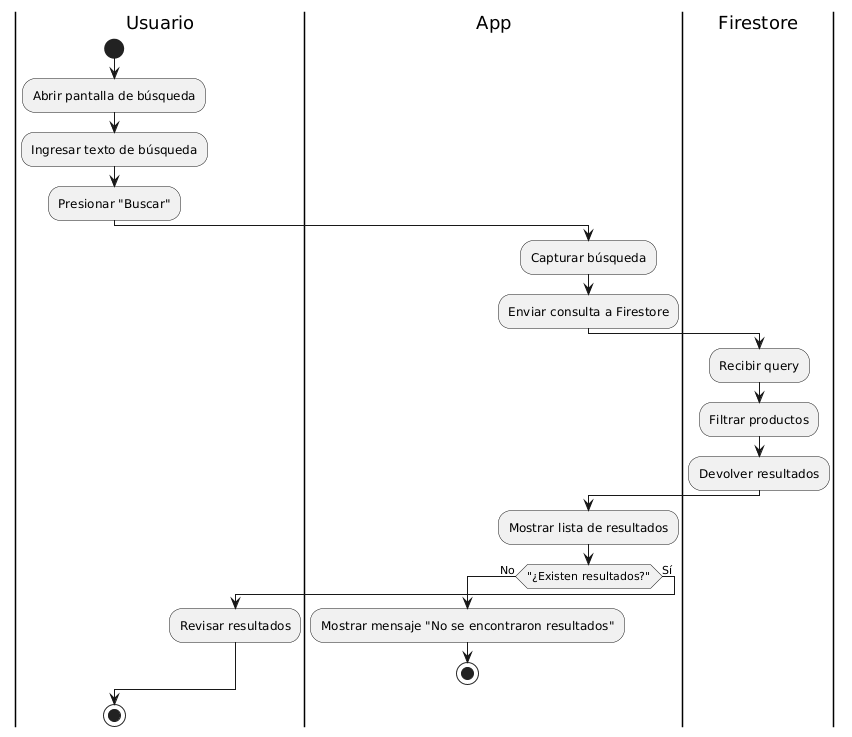
1. Registro Atomatico IA



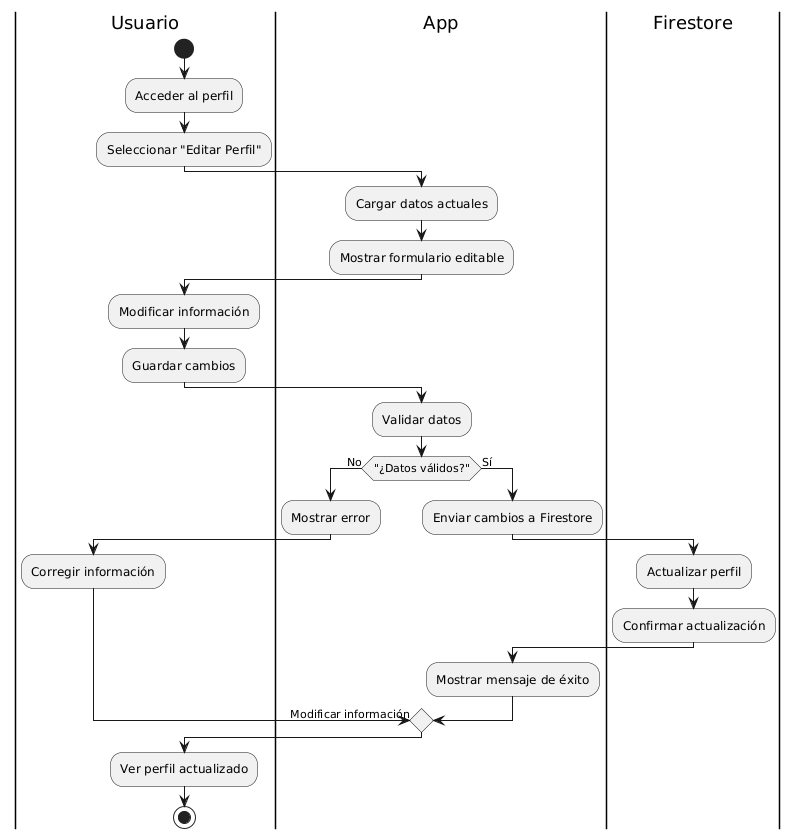
1. Crear Colección



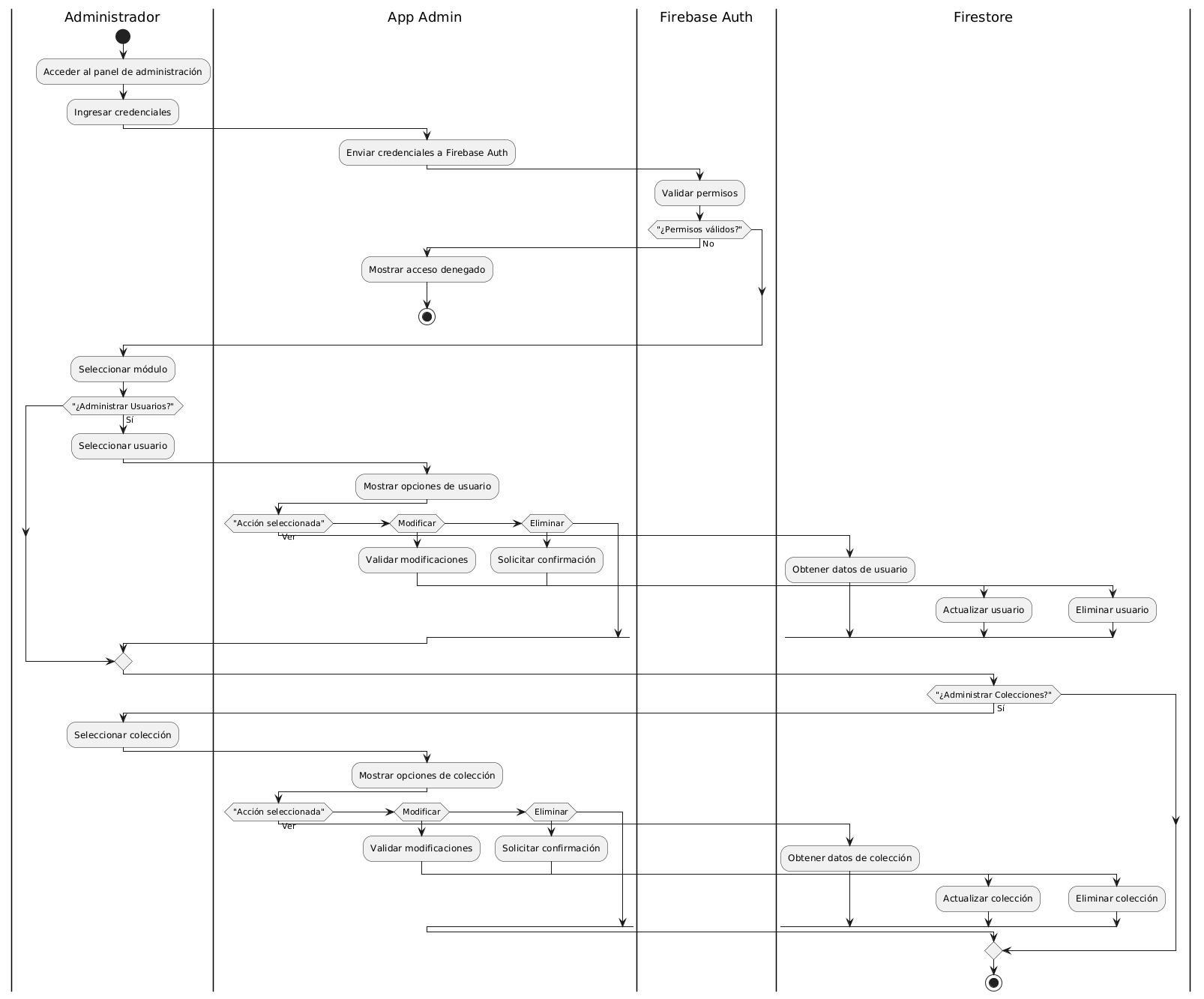
1. Filtrar Producto



1. Editar Perfil



1. Administración de sistema

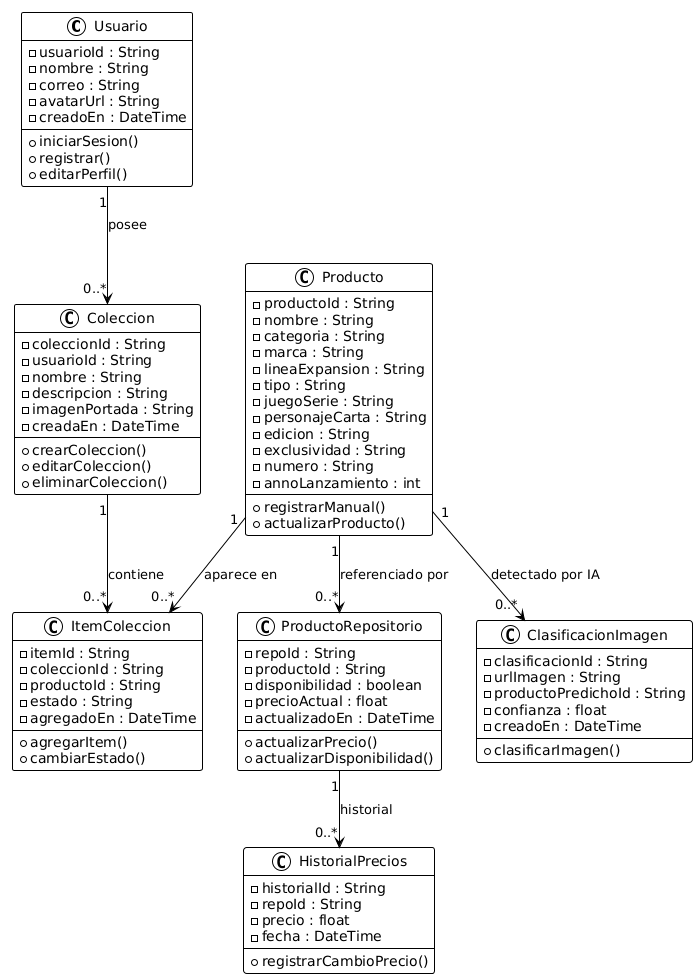


1. **Vista Lógica**

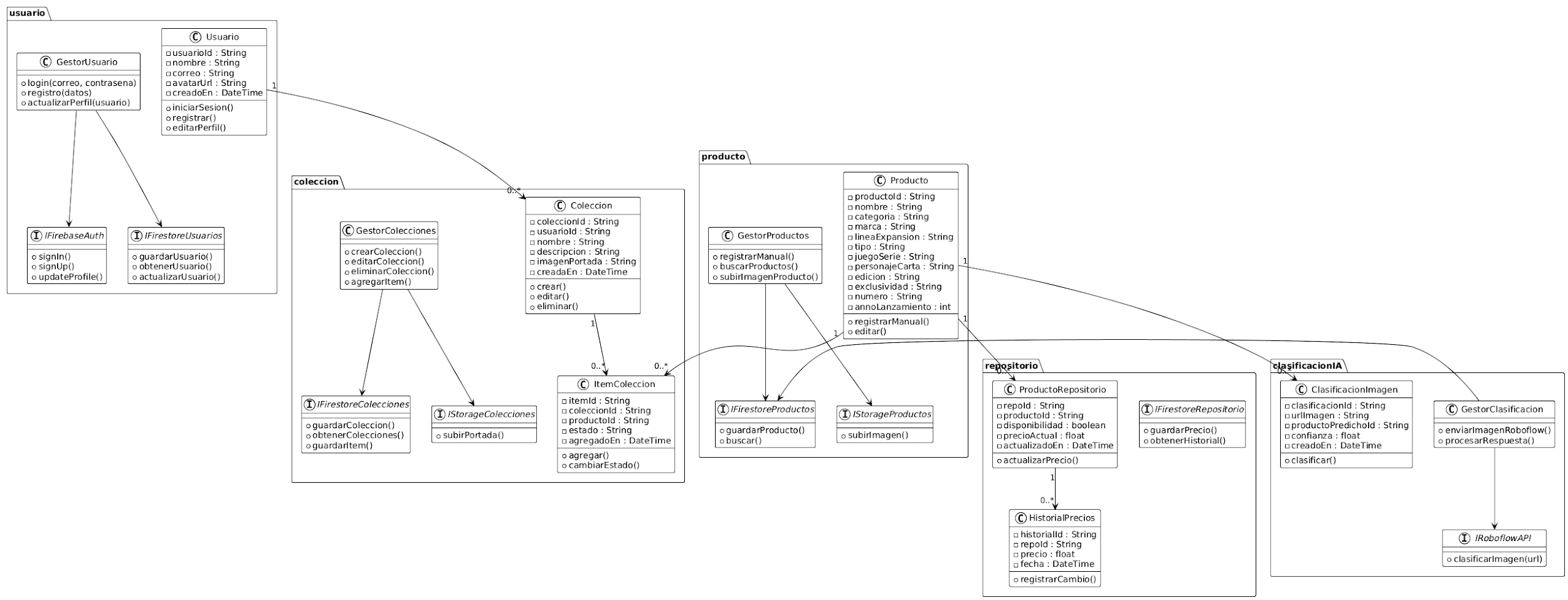
A continuación, se presenta una vista lógica de la aplicación expresado en tres diagramas, uno de ellos que muestra la parte estructural o estática de la aplicación (clases) y a la base de datos (modelo relacional). Otra vista que representa la parte dinámica (secuencias).

* 1. **Parte Estructural (Diagrama de Clases y Diagrama Relacional)**

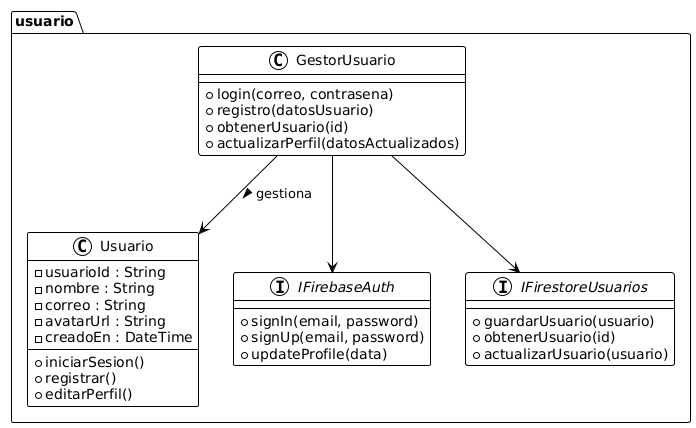
**Diagrama de Clases General**



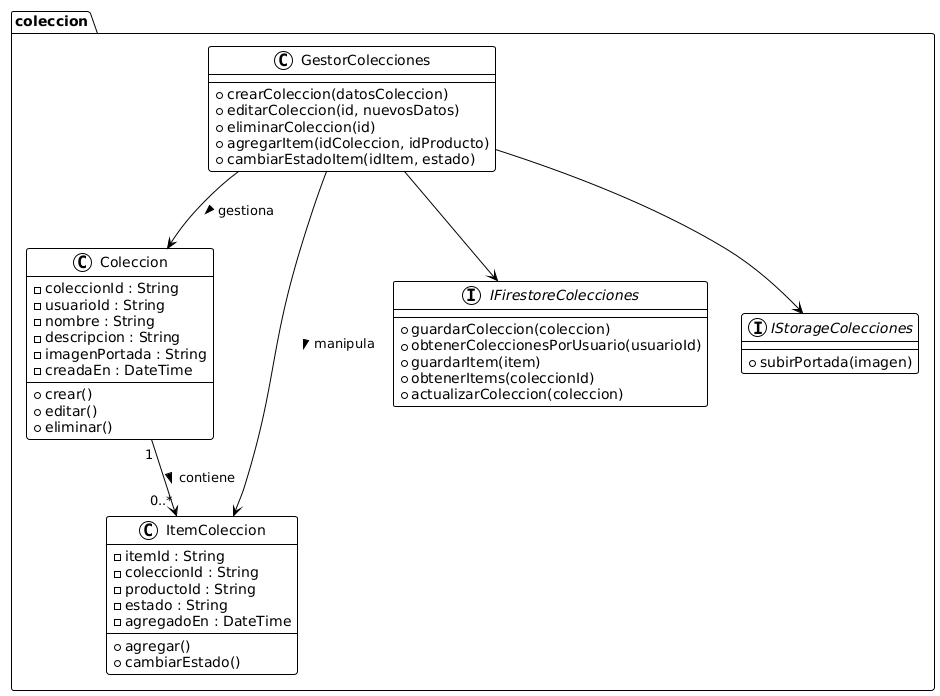
**Diagrama de Clases por paquete**



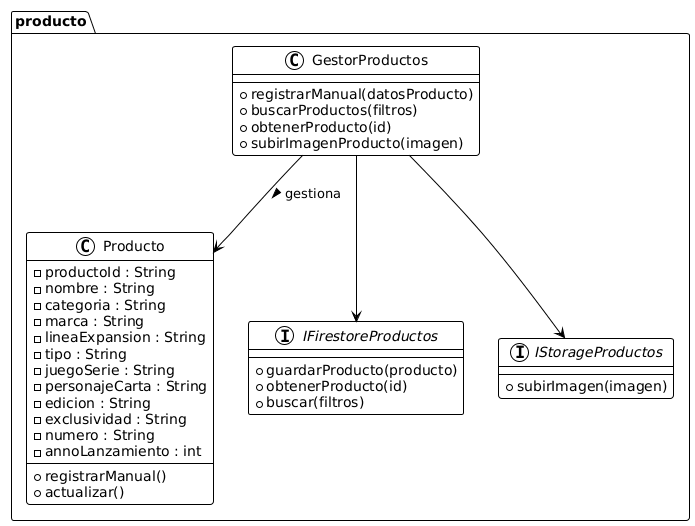
1. Diagrama de Clases: Usuario



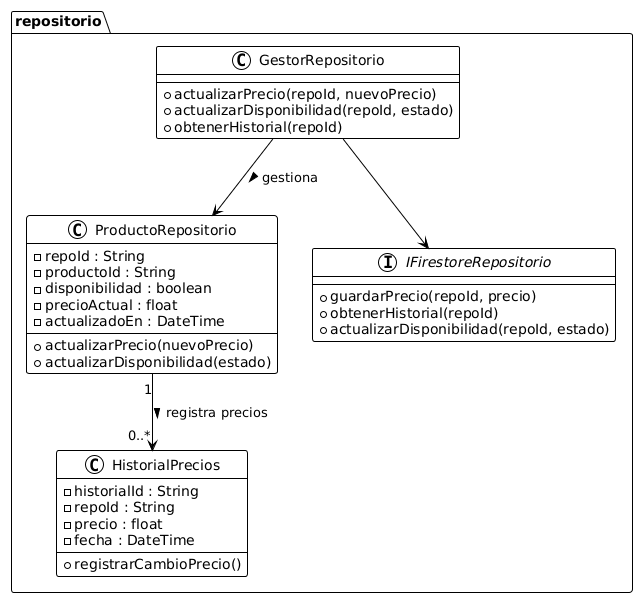
1. Diagrama de Clases: Colección



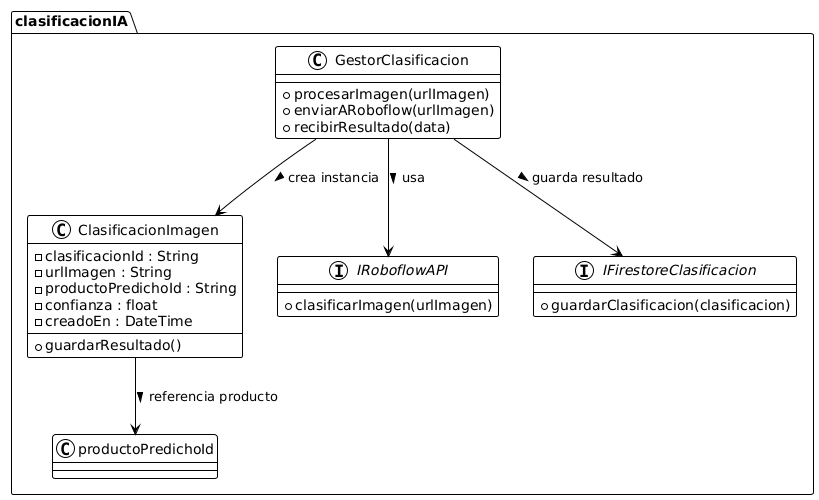
1. Diagrama de Clases: Producto



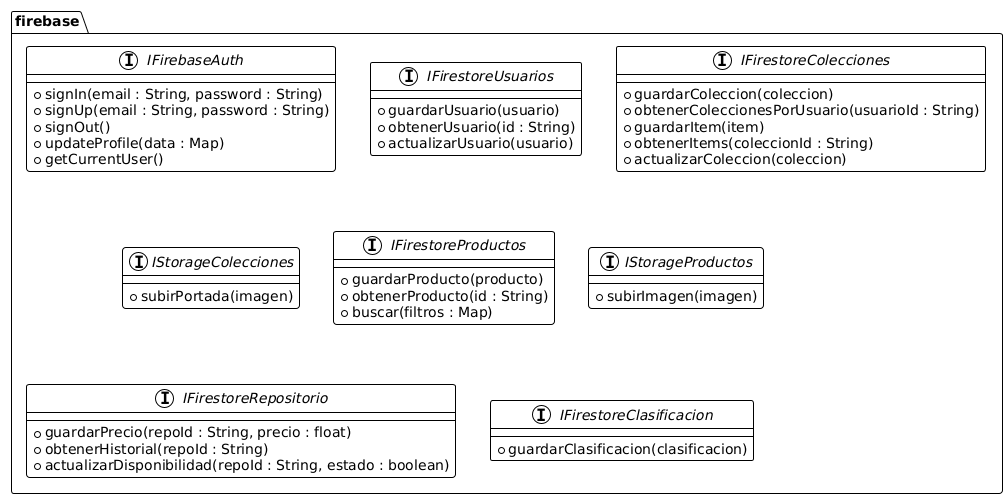
1. Diagrama de Clases: Paquete Repositorio



1. Diagrama de clases: Clasificación IA



1. Diagrama de Clase: Firebase

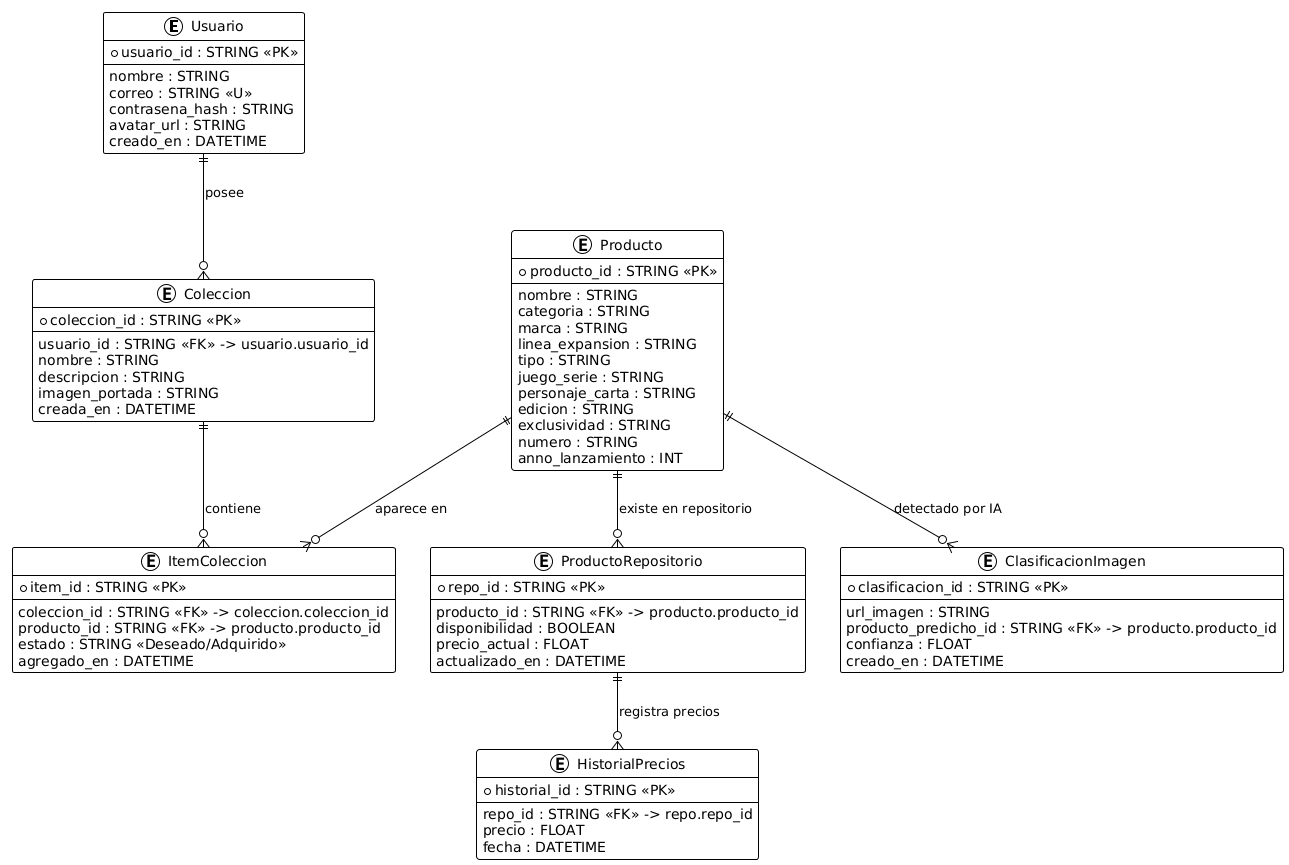


### Descripción de Clases

En la siguiente tabla se describen las principales clases identificadas en el modelo de dominio del sistema “Collecion APP”. Cada clase contiene atributos y comportamientos asociados a las funcionalidades del sistema, representadas en los diagramas de clases UML.

|  |  |  |
| --- | --- | --- |
| **Código** | **Nombre** | **Descripción** |
| CL-001 | Usuario | Representa a un usuario del sistema. Contiene su información personal (nombre, correo, avatar) y métodos para iniciar sesión y editar su perfil. |
| CL-002 | Gestor Usuario | Controla las operaciones relacionadas con usuarios: login, registro y actualización del perfil. |
| CL-003 | IFirebaseAuth | Interfaz para autenticación con Firebase (signIn, signUp, updateProfile). |
| CL-004 | IfirestoneUsuarios | Interfaz para manejar persistencia de datos de usuario en Firestore (guardar, obtener, actualizar). |
| CL-005 | Coleccion | Representa una colección creada por un usuario. Contiene nombre, descripción, imagen de portada y fecha de creación. |
| CL-006 | GestorColecciones | Controla la creación, edición, eliminación y administración general de colecciones. |
| CL-007 | ItemColeccion | Elemento individual dentro de una colección, asociado a un producto y con estado propio. |
| CL-008 | FirestoreColecciones | Interfaz para la persistencia de colecciones e items en Firestore. |
| CL-009 | IStorageColecciones | Interfaz encargada de subir y gestionar las imágenes de portada de las colecciones. |
| CL-010 | Producto | Representa un producto físico. Incluye atributos como categoría, marca, serie, exclusividad, número y año. |
| CL-011 | GestorProductos | Administra el registro manual de productos, la búsqueda y la gestión de imágenes. |
| CL-012 | IFirestoreProductos | Interfaz para almacenar y recuperar productos desde Firestore. |
| CL-013 | IStorageProductos | Interfaz para la subida y manejo de imágenes de productos. |
| CL-014 | ProductoRepositorio | Representa la información de mercado de un producto (precio actual, disponibilidad, fecha de actualización). |
| CL-015 | HistorialPrecios | Almacena los registros de cambios de precio asociados a un producto. |
| CL-016 | IFirestoreRepositorio | Interfaz para guardar y obtener historiales de precios desde Firestore. |
| CL-017 | ClasificacionImagen | Representa el resultado de la clasificación de una imagen por IA, incluyendo producto detectado y nivel de confianza. |
| CL-018 | GestorClasificacion | Gestiona el proceso de clasificación: envía imágenes al modelo IA y procesa la respuesta. |
| CL-019 | IRoboflowAPI | Interfaz que permite comunicarse con el modelo de IA para realizar la clasificación de imágenes. |
|  |  |  |
|  |  |  |

**Ilustración 4: Diagrama de Base Datos (Relacional)**

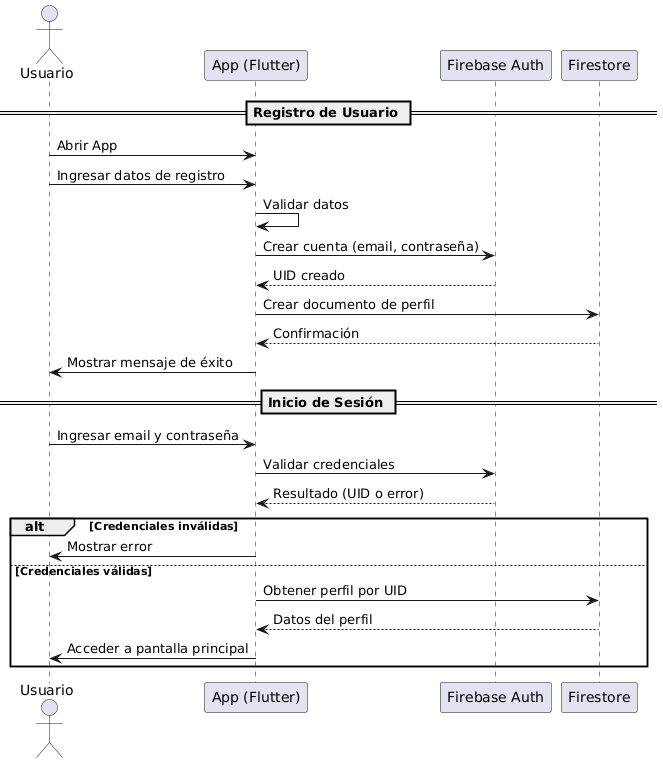


### Descripción de Tablas

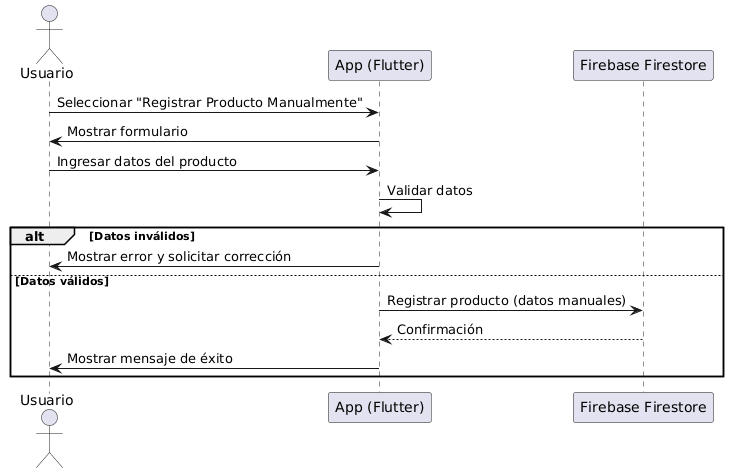
|  |  |  |
| --- | --- | --- |
| **Código** | **Nombre** | **Descripción** |
| TB-001 | Usuario | Almacena la información de los usuarios registrados en la aplicación, incluyendo datos personales, credenciales y configuración básica del perfil. Cada usuario puede crear y administrar múltiples colecciones. |
| TB-002 | Colección | Representa una colección creada por un usuario. Contiene meta-información como nombre, descripción, imagen de portada y la fecha de creación. Cada colección puede contener múltiples ítems pertenecientes a distintos productos. |
| TB-003 | Producto | Registra la información detallada de los productos coleccionables disponibles en la aplicación. Incluye atributos característicos como categoría, marca, línea o expansión, tipo, juego o serie, personaje o carta representada, edición, exclusividad, número del producto, año de lanzamiento y nombre oficial. Es la tabla base del catálogo referenciado por el repositorio y las colecciones de usuario. |
| TB-004 | ItemColecion | Asocia un producto específico con una colección creada por un usuario. Indica si el producto está deseado o adquirido, y registra la fecha en que se añadió a la colección. Funciona como tabla puente entre Colección y Producto. |
| TB-005 | ClasificaciónImagen | Almacena los resultados generados por la inteligencia artificial tras analizar imágenes enviadas por el usuario. Guarda la URL de la imagen, el producto predicho, el nivel de confianza del modelo y la fecha en que se realizó la clasificación. |
| TB-006 | HistorialPrecio | Registra los cambios históricos de precio asociados a los productos del repositorio. Permite realizar auditorías, comparaciones y análisis de variaciones de precio a lo largo del tiempo. |
| TB-007 | Item coleccion | Registra la pertenencia de un producto dentro de una colección específica del usuario, indicando además su estado (deseado o adquirido) y la fecha en que fue agregado. |
|  |  |  |
|  |  |  |

* 1. **Parte Dinámica (Diagrama de Secuencias)**

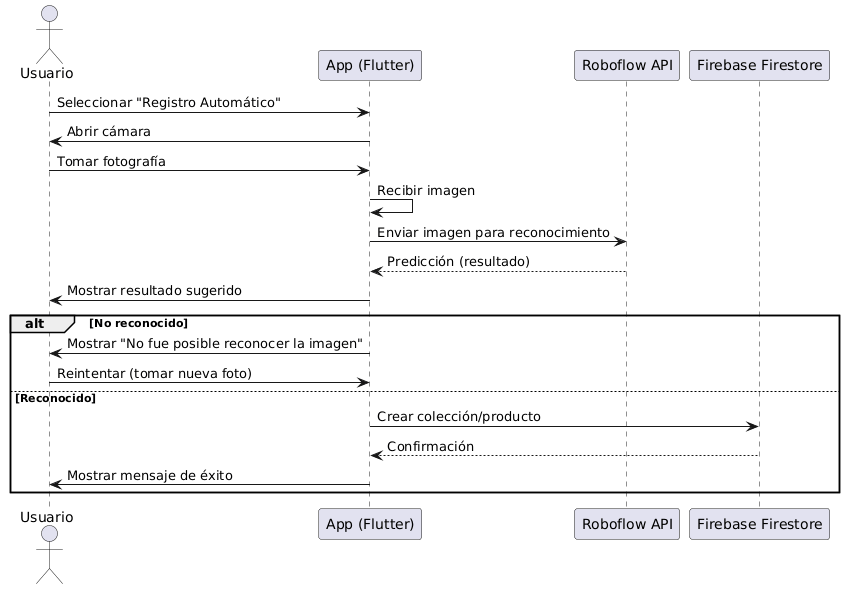
**Ilustración 1: Registro e Inicio de Sesión de Usuario**



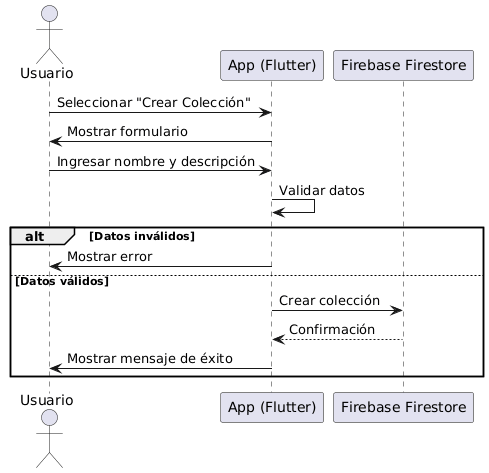
**Ilustración 2: Registro Manual de Producto**



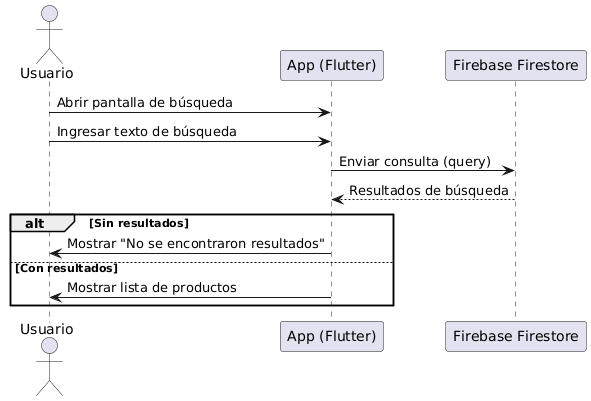
**Ilustración 3: Registro Automático**



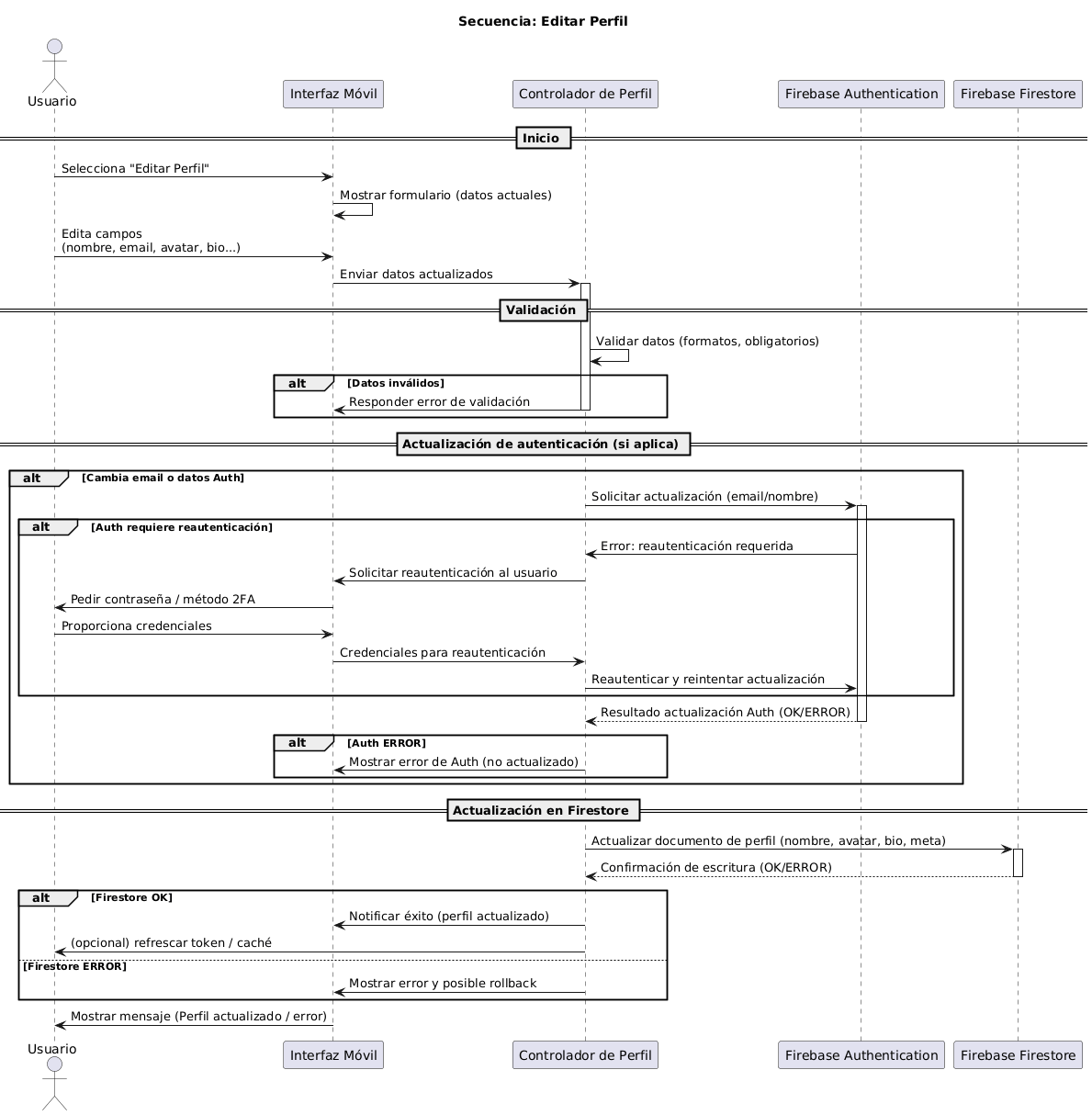
**Ilustración 4: Crear Colección**



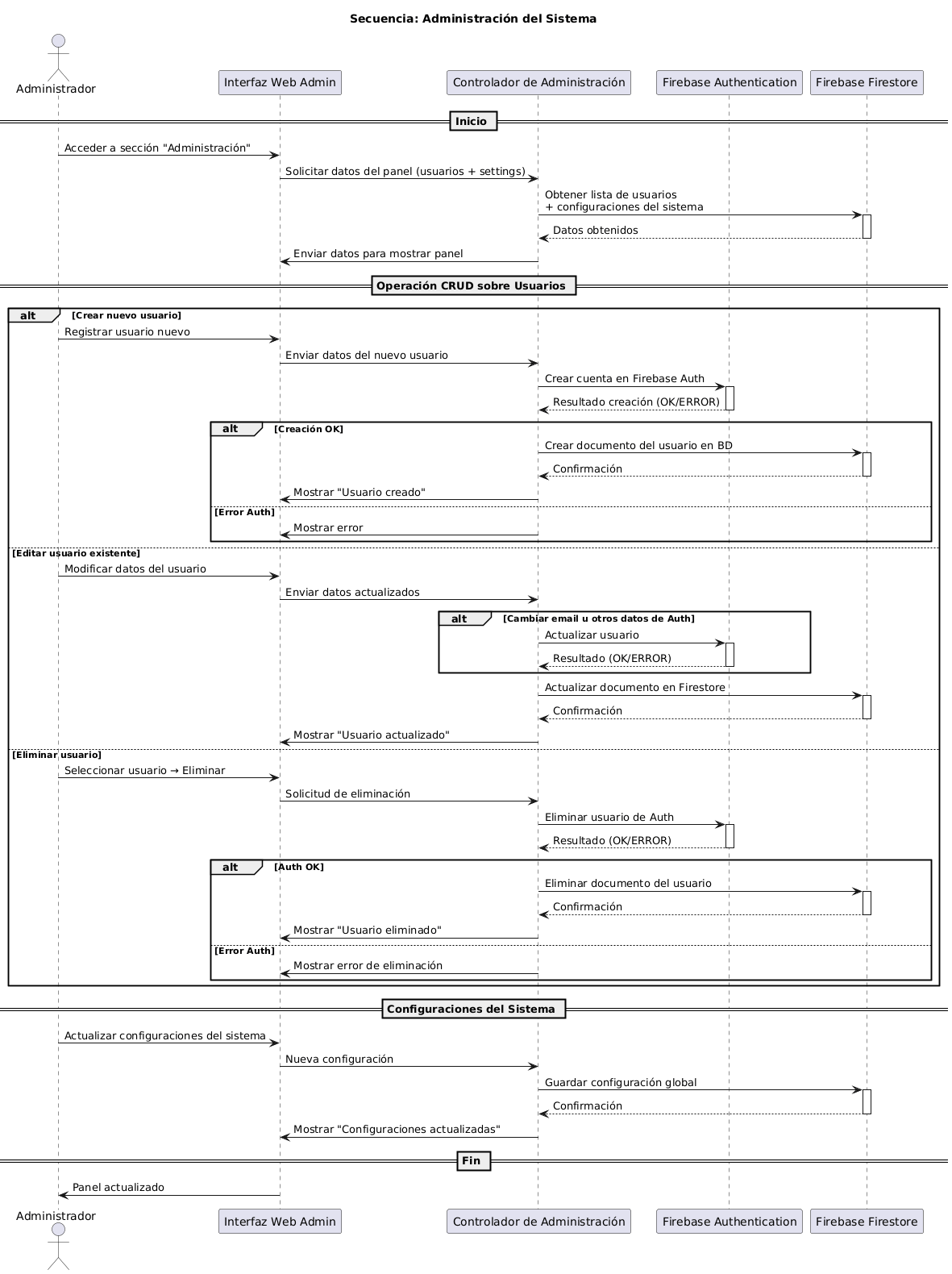
**Ilustración 5: Buscar Producto**



**Ilustración 6: Editar Perfil**



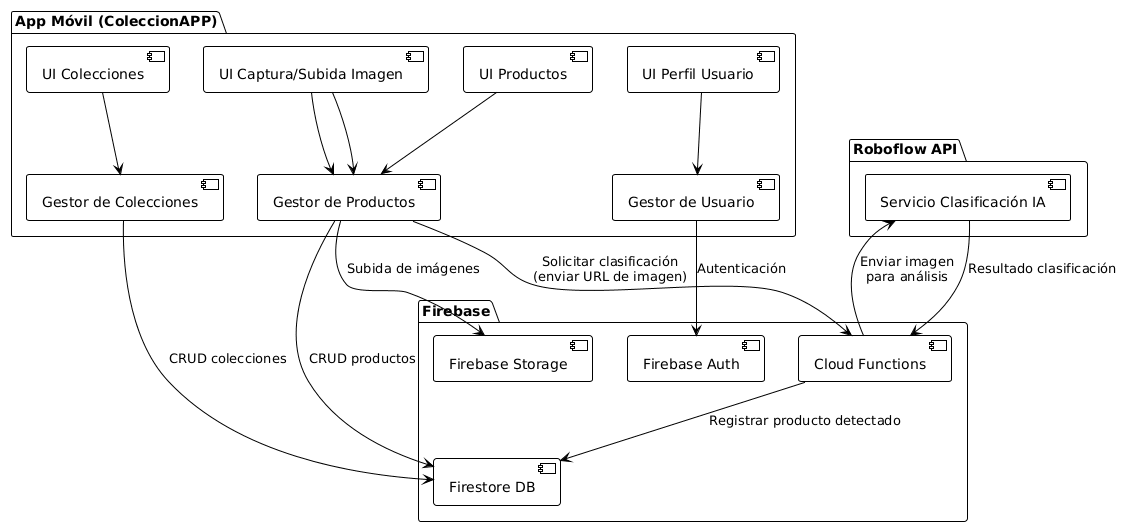
**Ilustración 7: Administrar Sistema**



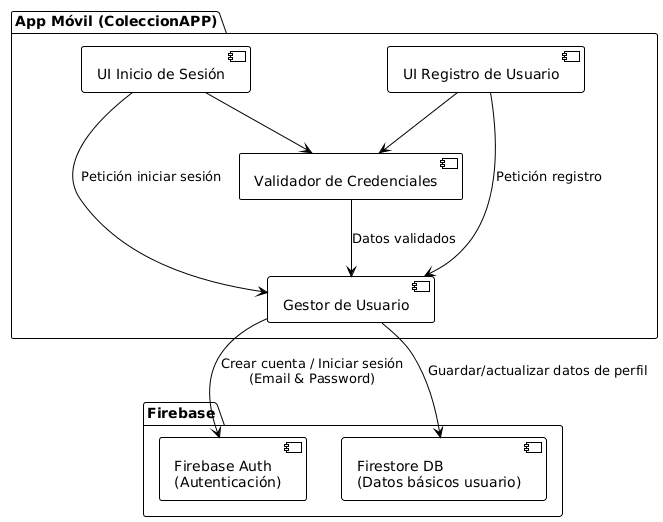
1. **Vista de Desarrollo o Despliegue**

En esta vista se describen las componente o modulos en las cuales se dividirá o implementará el sistema

**Ilustración 1: Diagrama componente general del sistema**



**Ilustración 2: Diagrama de componentes Inicio y Registro de Usuario**



**Ilustración 3: Diagrama de componentes Crear Colección**

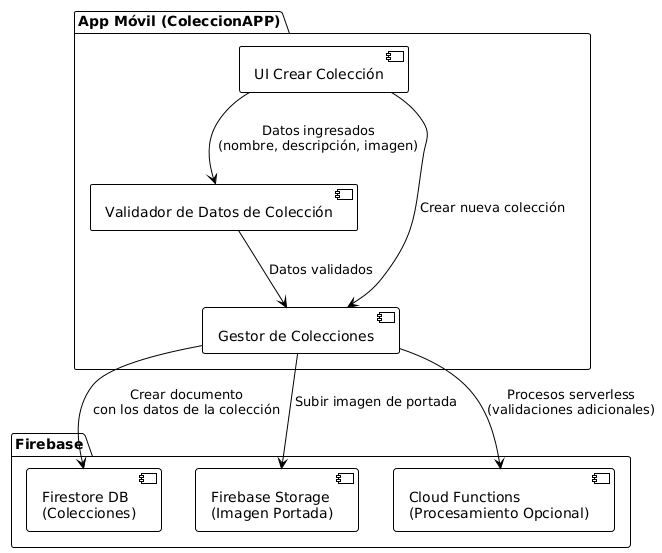


Ilustración 4: Registro Manual de Producto

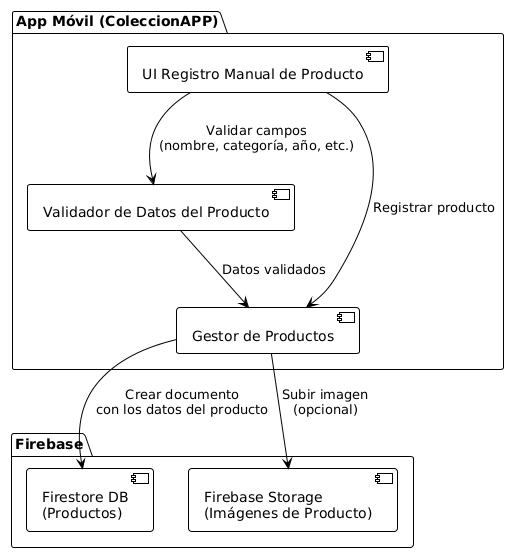


ilustración 5: Búsqueda de producto

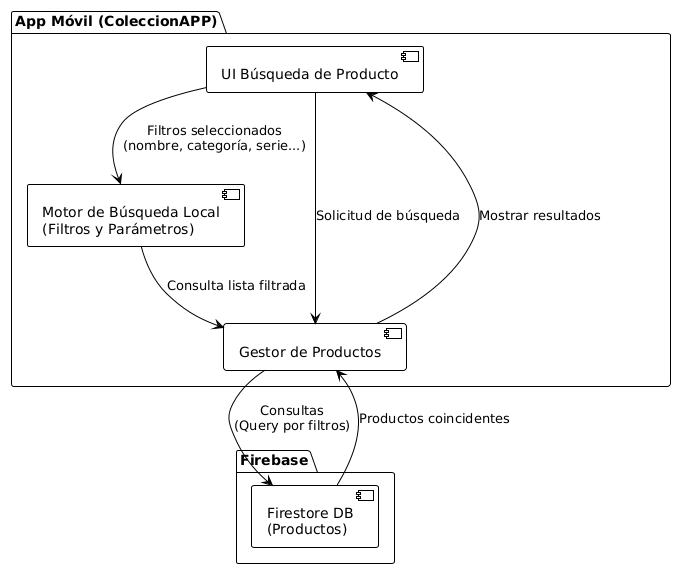


ilustración 6: diagrama de componentes Registro Automático de producto IA

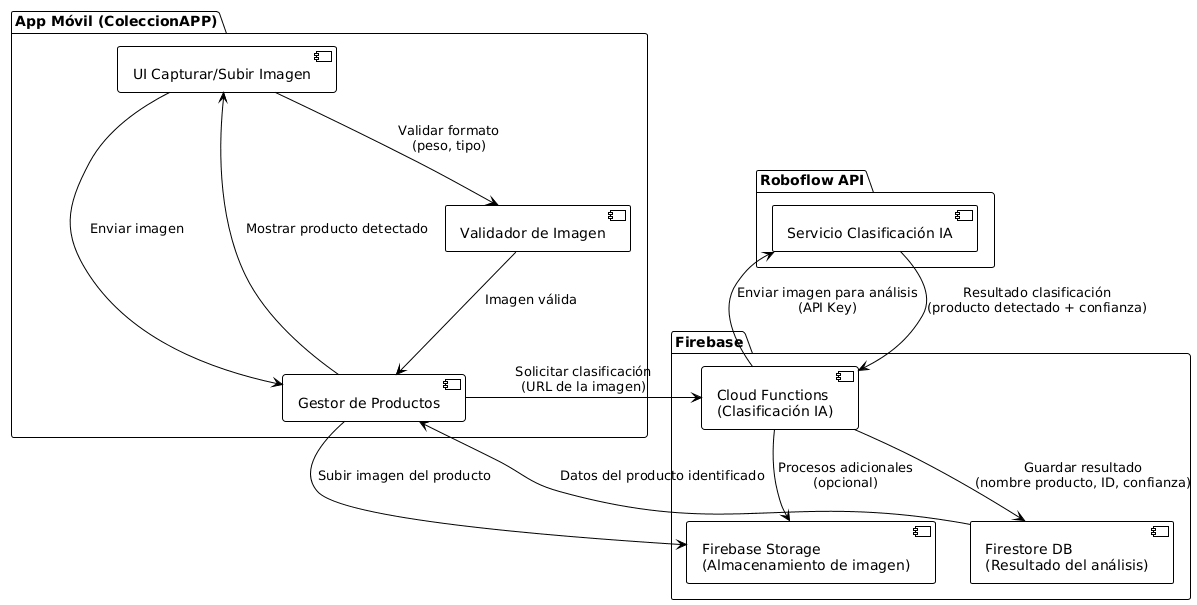
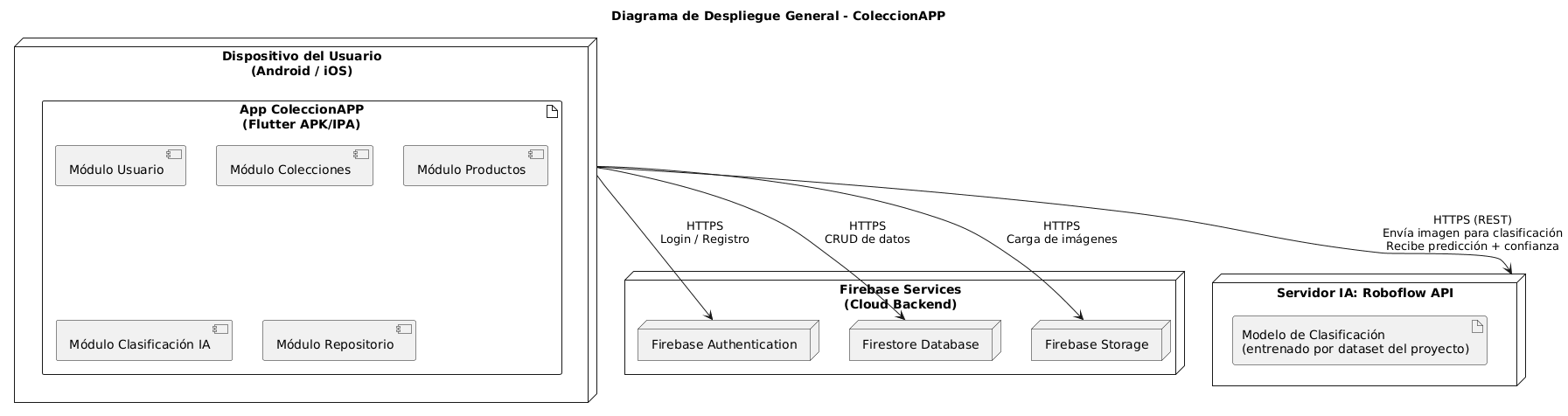


ilustración 7: diagrama de componentes Editar Perfil

**7. Vista Fisica**

La Vista de Escenarios permite describir los comportamientos clave del sistema ColeccionAPP frente a sus actores, así como los principales escenarios de calidad que influyen en las decisiones arquitectónicas. Esta vista se relaciona directamente con los casos de uso descritos en el ERS y muestra cómo el sistema responde a las necesidades funcionales y no funcionales priorizadas.

diagrama despliegue general del sistema



**8. Decisiones de Diseño y Selección de Alternativas**

Las decisiones arquitectónicas adoptadas para el desarrollo de **ColeccionAPP** se fundamentan en los requerimientos funcionales y no funcionales definidos en el ERS, así como en las restricciones de tiempo, infraestructura, portabilidad y mantenibilidad del proyecto. A continuación, se describen las decisiones más relevantes y las alternativas evaluadas.

**8.1 Justificación General de la Arquitectura Adoptada**

Se seleccionó una arquitectura **modular basada en servicios cloud y componentes desacoplados**, alineada con los siguientes factores:

* **Tiempo de construcción reducido (5 semanas)**, lo que obliga a usar herramientas de alta productividad y bajo mantenimiento.
* **Multiplataforma**: La aplicación debe ejecutarse en Android, iOS, Windows y macOS.
* **Dependencia mínima de backend tradicional**, priorizando soluciones serverless.
* **Uso obligatorio de IA para reconocimiento de imágenes**, utilizando la API de Roboflow.
* **Escalabilidad futura**: ampliación de repositorios, aumento de usuarios y posibilidad de agregar funciones sociales.

**8.2 Principales Decisiones de Diseño**

**Decisión 1: Framework Flutter + Dart para la aplicación móvil**

**Alternativas evaluadas:** React Native, Kotlin Multiplatform, desarrollo nativo.  
**Justificación:**

* Código único multiplataforma (Android, iOS, desktop).
* Mejor rendimiento gráfico para interfaces dinámicas.
* Integración nativa con Firebase.
* Permite entregar rápidamente un MVP funcional.

**Decisión 2: Firebase como Backend Serverless**

**Servicios seleccionados:**

* **Firebase Authentication**
* **Firestore Database**
* **Firebase Storage**

**Justificación:**

* En el ERS se establece dependencia con Google Services.
* Permite escalabilidad sin configurar servidores.
* Manejo seguro de credenciales.
* Soporta sincronización en tiempo real.
* Minimiza el desarrollo de backend.

**Alternativas evaluadas:** API REST propia en Node.js o Django, Supabase, AWS Amplify.

**Decisión 3: Servidor de IA externo (Roboflow API)**

**Justificación:**

* Se requiere reconocimiento de productos mediante visión artificial.
* Entrenamiento rápido de modelos con datasets propios.
* API lista para uso, compatible con Flutter vía HTTP.
* Satisface el requisito no funcional **R.7** (precisión ≥ 85%).

**Alternativas evaluadas:** TensorFlow Lite embebido, modelo ONNX local, Google Vision AI.  
**Razón de descarte:** consumo elevado de recursos, menor precisión o complejidad de implementación.

**Decisión 6: Uso de Detección Automática (IA) + Registro Manual**

**Razón:**

* Apoya el flujo crítico del proyecto definido en el ERS:  
  *Registrar productos con foto (IA) o manualmente.*

**Justificación técnica:**

* La IA puede fallar o no reconocer algunos productos.
* El flujo manual garantiza que siempre exista un mecanismo alternativo.

**Decisión 7: Uso de Firestore como BD no relacional**

**Razones:**

* Los datos del sistema (productos, colecciones, usuario, historial de precios) son altamente jerárquicos.
* Workflows basados en documentos y colecciones son ideales.

**Alternativas:** PostgreSQL, MySQL.  
**Razón de descarte:** requerirían backend adicional.

**8.3** **Riesgos Arquitectónicos Identificados**

|  |  |
| --- | --- |
| **Riesgo** | **Mitigación** |
| Dependencia de Roboflow | Documentar fallback para registro manual; prever cambio futuro a otro proveedor IA. |
| Límite de Firestore en operaciones concurrentes | Optimizar consultas y estructura; usar índices; paginación. |
| IA con precisión inferior a 85% | Ajustar dataset, aumentar entrenamiento, filtrar imágenes de entrada. |

8**.4 Conclusión:**

Las decisiones arquitectónicas aquí descritas permiten cumplir eficientemente con los requerimientos del ERS, garantizando que **ColeccionAPP** sea:

* Mantenible
* Escalable
* Rápido de implementar
* Seguro
* Multiplataforma
* Compatible con IA y servicios en la nube

y preparado para futuras extensiones, como funcionalidades sociales, paneles avanzados, dashboards y ampliación del repositorio de productos.

**9. Análisis de Reutilización**

El sistema ColeccionAPP fue diseñado considerando la posibilidad de reutilizar componentes, módulos y servicios en futuras versiones del proyecto o en aplicaciones relacionadas, mejorando competitividad, reduciendo tiempos de desarrollo y disminuyendo costos de mantenimiento.  
A continuación, se detalla el análisis de reutilización desde los niveles técnico, arquitectónico y funcional.

**9.1 Reutilización a nivel de Arquitectura**

La arquitectura del sistema sigue un enfoque modular y desacoplado, lo que permite reutilizar bloques completos del sistema en futuras implementaciones:

Componentes reutilizables

Autenticación (Firebase Auth):  
Puede extenderse para otras aplicaciones móviles que requieran login por email o Google sin modificar la lógica base.

Servicios de Base de Datos (Firestore):  
Los modelos de datos de *Usuario*, *Colección*, *Producto* e *ItemColección* están normalizados y pueden integrarse en aplicaciones que manejen catálogos o inventarios.

Gestores de Dominio (GestorUsuario, GestorColecciones, GestorProductos):  
Estos módulos encapsulan lógica de negocio y pueden ser reusados o extendidos sin alterar la interfaz pública.

Módulo de Clasificación por IA:  
El componente IRoboflowAPI + GestorClasificacion es completamente desacoplado y puede:

reutilizarse con otro dataset,

apuntar a otro modelo de Roboflow,

migrarse a otro proveedor de IA sin cambios complejos.

**9.2 Reutilización en el Código Fuente**

La aplicación emplea Dart + Flutter, lo que permite:

1. Reutilización de Widgets y Vistas

* Formularios reutilizables (login, registro, edición).
* Tarjetas y celdas de producto.
* Grillas de colecciones.
* Componentes visuales reutilizables:  
  Validadores, alertas, pop-ups, listas, botones, temas.

2. Reutilización de Servicios

* Servicios de autenticación.
* Servicios de almacenamiento de imágenes en Firebase Storage.
* Servicios de consultas y búsquedas sobre Firestore.
* Manejadores de errores y logs.

3. Reutilización del flujo de IA

* El flujo completo *Captura → Envío → Clasificación → Validación → Registro* puede insertarse en:
* Apps futuras de inventario
* Apps de catálogo
* Apps de reconocimiento visual para ventas/mercado secundario

**9.3 Reutilización de Reglas de Negocio**

Las reglas de negocio definidas en la aplicación son genéricas, por ejemplo:

Validación de creación de colecciones.

Políticas de edición/eliminación.

Estados del producto (Deseado / Adquirido).

Control de permisos (usuario con sesión válida).

Validaciones de formato y longitud en formularios.

Todo esto puede reutilizarse en:

Otras apps de coleccionismo,

Apps de inventario personal,

Sistemas que manejan repositorios de datos categorizados.

**9.4 Reutilización de Datos**

Los siguientes datasets pueden ser utilizados por versiones futuras del proyecto o sus componentes:

Repositorio de productos actualizable:

Puede extenderse para nuevas colecciones, marcas o líneas de productos.

Historial de precios:  
Puede servir para análisis comparativo, dashboards o funcionalidades de valorización futura.

Dataset entrenado para IA:  
Reutilizable para:

retrain incremental,

mejora del modelo,

creación de API propia de clasificación,

un modelo de clasificación multiplataforma.

**9.5 Beneficios de la Reutilización**

Reducción de esfuerzo en futuros proyectos:

Gran parte del backend y lógica ya está implementada.

Disminución de tiempo de desarrollo:  
Se pueden reutilizar módulos críticos como autenticación, manejo de colecciones e IA.

Claridad en la evolución del sistema:  
La arquitectura modular facilita añadir nuevas funcionalidades sin rupturas.

Ahorro de costos:  
La reutilización reduce costos de pruebas, diseño y mantenimiento.

**9.6 Conclusión del Análisis de Reutilización**

El diseño modular de ColeccionAPP permite una reutilización amplia tanto de los componentes de software como de los modelos de datos y arquitecturas.  
Su implementación basada en Flutter + Firebase + Roboflow facilita extender la aplicación a nuevas plataformas, aprovechar servicios existentes y escalar el sistema con bajo costo.

La reutilización ha sido incluida como un eje arquitectónico estratégico, coherente con el ERS y con las metas del proyecto descritas en el DAS.