Arquitectura del problema: ¿Quién es tu papá?

Presentado por:

Jhon Mario Diaz

Juan David Diaz

Andrés Felipe Rico

Presentado a:  
Julio Cesar



Universidad San Buenaventura Cali

Facultad de ingeniería

Programa de ingeniería de sistemas

Valle del cauca, Cali

2025

1. Índice

|  |
| --- |
| [1. Introducción 2](#_Toc1723917325)  [1.1 Objetivos y alcance de este documento 3](#_Toc1873189186)  [1.2 Objetivos arquitectónicos y restricciones 3](#_Toc462351299)  [1.2.1 Objetivos arquitectónicos 3](#_Toc923458033)  [1.2.2 Restricciones 4](#_Toc1112752822)  [2. Requerimientos funcionales (RF) 5](#_Toc966922605)  [3. Requerimientos No funcionales (RNF) 6](#_Toc2018769585)  [4. Atributos de calidad 8](#_Toc2107380724)  [5. Identificación de interesados 9](#_Toc1141356388)  [6. Interesados VS Requerimientos No Funcionales (Ponderación) 10](#_Toc559036080)  [7. Atributos de calidad más importante (Drivers arquitectónicos) 10](#_Toc257195522)  [8. Identificación de las tácticas para atacar los atributos de calidad 12](#_Toc1246800590)  [9. Patrones arquitectónicos que a utilizar 15](#_Toc2095300165)  [10. Estrategia (Táctica con los Patrones Seleccionados) 16](#_Toc562088219)  [11. Modelo 4 + 1 19](#_Toc1429316405)  [11.1 Escenarios 19](#_Toc1000009884)  [Casos de Uso 20](#_Toc1161133780)  [11.2 Vista Lógica 21](#_Toc510879978)  [11.3 Vista de Implementación 21](#_Toc1865999973)  [API Gateway & Load Balancer 22](#_Toc160534891)  [Servicios de Microservicios 22](#_Toc456398818)  [Almacenamiento de Datos 22](#_Toc2051156279)  [Servicios de Infraestructura 23](#_Toc337268671)  [11.4 Vista de Procesos 23](#_Toc284892726)  [Proceso de Autenticación 24](#_Toc574512241)  [Proceso de Búsqueda 24](#_Toc1779360619)  [Proceso de Adición de Familiar 24](#_Toc2022820843)  [11.5 Vista Física 24](#_Toc158113408)  [Por cada región: 25](#_Toc654036885)  [Capa de Red y Seguridad 25](#_Toc1300135180)  [Almacenamiento 25](#_Toc1304503618)  [Mensajería y Monitoreo 25](#_Toc1750508761)  [11.6 Relación con Atributos de Calidad Prioritarios 25](#_Toc705010943)  [1. Fiabilidad 26](#_Toc1106713844)  [2. Seguridad 26](#_Toc1020598451)  [3. Eficiencia de Desempeño 26](#_Toc2023927394)  [4. Escalabilidad 26](#_Toc1176788729)  [5. Idoneidad Funcional 26](#_Toc917272154) |

# Introducción

El presente documento describe la arquitectura de software para el sistema "¿Quién es tu papá?" (nombre original: "Who’s Your Daddy?"), una plataforma innovadora diseñada para construir y gestionar el árbol genealógico más extenso y detallado del mundo. El objetivo fundamental de esta arquitectura es soportar una base de millones de usuarios y cientos o miles de aplicaciones de terceros, facilitando el descubrimiento, preservación y compartición de la historia familiar.

La visión arquitectónica se centra en crear un sistema robusto, escalable y seguro, capaz de manejar la naturaleza dinámica y el crecimiento exponencial de los datos genealógicos. Para ello, se priorizan atributos de calidad críticos como la disponibilidad, escalabilidad, rendimiento, seguridad e interoperabilidad, materializados a través de una arquitectura de microservicios desplegada en la nube.

## Objetivos y alcance de este documento

Este documento sobre la arquitectura del problema ¿Quién es tu papa? tiene como objetivos:

* Describir las decisiones arquitectónicas clave que guiarán el diseño, desarrollo y evolución del sistema "¿Quién es tu papá?".
* Comunicar la estructura y el comportamiento de alto nivel del sistema a todos los interesados, incluyendo equipos de desarrollo, operaciones, seguridad y gestión.
* Asegurar que el diseño arquitectónico satisface los requisitos funcionales y no funcionales (atributos de calidad) identificados.
* Servir como base para futuras revisiones y evoluciones de la arquitectura.

El alcance de este documento abarca las principales vistas arquitectónicas (Lógica, de Implementación, de Procesos, Física y de Escenarios/Casos de Uso), los patrones seleccionados, las tácticas para abordar los atributos de calidad y las tecnologías clave

## Objetivos arquitectónicos y restricciones

## Objetivos arquitectónicos

Los siguientes objetivos arquitectónicos son fundamentales para el éxito del sistema "¿Quién es tu papá?" y derivan directamente de los requisitos de negocio y los atributos de calidad priorizados:

* **Escalabilidad Masiva:** La arquitectura debe ser capaz de gestionar de manera eficiente un volumen de datos genealógicos en constante crecimiento, representando miles de millones de nodos y relaciones, y soportar millones de usuarios activos concurrentes y miles de aplicaciones de terceros.
* **Alto Rendimiento en Consultas Complejas:** El sistema debe ofrecer tiempos de respuesta bajos, incluso para búsquedas complejas en el grafo genealógico (p. ej., "todos los hombres de 80 años" en < 3000 ms) y procesar 10,000 RPS en picos de demanda.
* **Máxima Fiabilidad e Integridad de Datos:** Se debe asegurar la consistencia del grafo genealógico y la precisión de los registros históricos, implementando mecanismos de recuperación automática y validación dual.
* **Seguridad Robusta y Privacidad:** Proteger la información personal sensible de acuerdo con normativas como GDPR/CCPA es imperativo, implementando cifrado en tránsito y en reposo, y controles de acceso granulares.
* **Interoperabilidad Extensiva:** Facilitar la integración con sistemas externos y el desarrollo de aplicaciones por terceros mediante una API pública bien definida, segura y completa.
* **Mantenibilidad y Evolución Ágil:** La arquitectura debe ser modular para permitir la corrección rápida de errores (MTTR objetivo: [especificar tiempo, ej: < 4 horas]) y la adición de nuevas funcionalidades o fuentes de datos con un impacto mínimo en el sistema existente.

## Restricciones

Las siguientes restricciones han influido en las decisiones arquitectónicas:

* **Naturaleza de los Datos:** Los datos son inherentemente un grafo de gran tamaño y complejidad, con relaciones no predefinidas que deben ser consultables eficientemente.
* **Fuentes de Datos Diversas:** El sistema debe integrar datos de múltiples fuentes, incluyendo entradas manuales, importaciones de formatos estándar (GEDCOM, CSV), APIs (Facebook) y registros históricos digitalizados.
* **Verificación Humana:** La necesidad de verificación dual y arbitraje de registros históricos impone requisitos en los flujos de trabajo y la gestión de datos.
* **Ecosistema Tecnológico (Implícito):** La elección de servicios de AWS (Neptune, DynamoDB, S3, Lambda, etc.) implica una preferencia por soluciones gestionadas en la nube para agilizar el desarrollo y la escalabilidad.
* **Regulatorias:** Cumplimiento estricto con GDPR, CCPA y otras posibles regulaciones sobre la privacidad y manejo de datos personales y genealógicos.
* **Organizacionales (A ser completado por el equipo):**
  + Presupuesto*:* [Especificar limitaciones presupuestarias si afectan las elecciones tecnológicas o de personal].
  + Habilidades del Equipo*:* [Especificar las competencias existentes o las necesidades de formación].
  + Cronograma del Proyecto*:* [Especificar hitos principales que puedan influir en la arquitectura o el phasing].

# Requerimientos funcionales (RF)

1. **RF-01: Gestión de personas**
   * **Detalle:** El sistema debe permitir registrar, editar y eliminar personas, así como gestionar sus relaciones familiares (padre, madre, hijos, cónyuge, etc.) dentro del árbol genealógico.
   * **Interesados:** Usuarios finales, Historiadores, Terceros (Apps externas), Product Manager (PM).
2. **RF-02: Importación de datos externos**
   * **Detalle:** El sistema debe ofrecer la capacidad de importar datos desde plataformas externas como Facebook, archivos en formato GEDCOM, archivos CSV y APIs públicas de registros históricos.
   * **Interesados:** Usuarios finales, Historiadores, Equipo IT.
3. **RF-03: Búsqueda avanzada**
   * **Detalle:** El sistema debe proporcionar una funcionalidad de búsqueda avanzada y global que permita a los usuarios encontrar personas por nombres, apellidos, fechas de nacimiento/muerte y tipo de parentesco. Esta búsqueda debe incluir un autocompletado inteligente.
   * **Interesados:** Usuarios finales, Apps externas, Historiadores.
4. **RF-04: Visualización del árbol**
   * **Detalle:** El sistema debe ofrecer una visualización interactiva del árbol genealógico con capacidad de zoom. Debe permitir aplicar filtros por ramas familiares y ofrecer una navegación sencilla y clara tanto en plataformas web como móviles.
   * **Interesados:** Usuarios finales, Equipo IT, Equipo UX/UI.
5. **RF-05: API abierto**
   * **Detalle:** El sistema debe proporcionar una API pública (REST) accesible para que terceros puedan desarrollar aplicaciones. Esta API debe ofrecer capacidades completas de lectura y escritura sobre la información genealógica, respetando los permisos y la seguridad.
   * **Interesados:** Terceros (Apps externas), Equipo IT, Equipo de Seguridad, Desarrolladores externos.
6. **RF-06: Verificación dual de registros**
   * **Detalle:** Los registros históricos que se ingresen al sistema (como censos, certificados civiles o religiosos) deben ser digitalizados y posteriormente validados mediante un sistema de arbitraje, preferiblemente automático o con intervención humana dual, para asegurar la calidad de los datos.
   * **Interesados:** Historiadores, Equipo de Seguridad, Equipo QA, Product Manager (PM).
7. **RF-07: Gestión de autenticación**
   * **Detalle:** El sistema debe implementar un mecanismo seguro de autenticación y gestión de cuentas de usuario. Esto incluye la posibilidad de integración con proveedores de identidad social (como Facebook, Google) y un sistema de credenciales propias (email y contraseña).
   * **Interesados:** Usuarios finales, Equipo IT, Equipo de Seguridad.
8. **RF-08: Moderación y control de versiones**
   * **Detalle:** El sistema debe proporcionar herramientas para la moderación de la información ingresada y un sistema de control de versiones que permita la revisión de cambios históricos. También debe ofrecer la posibilidad de deshacer acciones realizadas sobre la información del árbol genealógico.
   * **Interesados:** Equipo de Seguridad, Usuarios finales, Historiadores, Product Manager (PM).

# Requerimientos No funcionales (RNF)

Los Requerimientos No Funcionales (RNF) definen los atributos de calidad y las restricciones bajo las cuales el sistema "¿Quién es tu papá?" debe operar. Estos son cruciales para asegurar que el sistema no solo cumpla con sus funcionalidades, sino que también lo haga de manera eficiente, segura y usable. A continuación, se detallan los RNF identificados:

* **RNF-01: Idoneidad funcional:** El sistema debe cubrir correctamente la gestión de personas y sus relaciones en el grafo genealógico. Esto incluye el registro, edición y búsqueda por parentesco, así como la importación de datos desde formatos como GEDCOM y plataformas como Facebook. La visualización del árbol debe ser infinita y clara. Los interesados principales en este requerimiento son los usuarios finales, historiadores y desarrolladores externos.
* **RNF-02: Eficiencia de desempeño:** Las consultas al grafo, como por ejemplo la búsqueda de "todos los bisabuelos", deben completarse en menos de 300 milisegundos (percentil 95). Adicionalmente, el sistema deberá ser capaz de procesar hasta 10,000 RPS (Requests Per Second) durante picos de demandas simultáneas. Este rendimiento es vital para los usuarios finales y el equipo de IT/Operaciones.
* **RNF-03: Compatibilidad:** Se requiere una integración fluida y sin conflictos ni pérdida de datos con la API de Facebook, el formato GEDCOM, archivos CSV y diversos navegadores y plataformas (web, PWA, iOS, Android). Los desarrolladores externos y el equipo de IT/Operaciones son los principales interesados.
* **RNF-04: Usabilidad:** El flujo de usuario para acciones comunes como agregar o buscar un familiar no debe requerir más de 3 clics. La interfaz de usuario debe ser clara para la visualización de ramas familiares, ofrecer un zoom infinito y filtros intuitivos tanto en la versión web como en las aplicaciones móviles. Este requerimiento es fundamental para los usuarios finales y el equipo de UX/UI.
* **RNF-05: Fiabilidad:** El sistema debe mantener la consistencia del grafo genealógico incluso después de caídas o fallos parciales, garantizando una recuperación automática sin corrupción de datos. También debe ofrecer tolerancia a desconexiones de red durante las ediciones realizadas por los usuarios. El equipo de IT/Operaciones, los historiadores y el equipo de QA son los principales interesados.
* **RNF-06: Seguridad:** Se debe garantizar la protección de los datos genealógicos de acuerdo con normativas como GDPR/CCPA. Esto incluye el cifrado de datos en tránsito (TLS) y en reposo (at-rest), un control de acceso por niveles (público/privado) y un registro de auditoría (audit trail) para los cambios críticos en la información. Los principales interesados son el equipo de Seguridad y los usuarios finales.
* **RNF-07: Mantenibilidad:** La arquitectura del sistema debe ser modular, preferiblemente basada en microservicios, para facilitar las actualizaciones y la corrección de errores. Se implementará un pipeline de Integración Continua y Despliegue Continuo (CI/CD) con pruebas automatizadas, especialmente para la ingestión de nuevos tipos de fuentes históricas. El equipo de IT/Operaciones, QA y el Product Manager (PM) son los interesados clave.
* **RNF-08: Portabilidad:** Los clientes del sistema (SPA, PWA, iOS, Android) deben compartir la mayor cantidad de lógica posible. El backend debe ser diseñado para ser migrable entre diferentes proveedores de nube (como AWS, Azure, GCP) sin necesidad de un retrabajo significativo. Este aspecto es importante para el equipo de IT/Operaciones y los desarrolladores externos.

# Atributos de calidad

Los atributos de calidad son las propiedades medibles del sistema que indican cuán bien satisface las necesidades de sus interesados. Estos atributos son esenciales para guiar las decisiones arquitectónicas. Para el sistema "¿Quién es tu papá?", se han identificado los siguientes atributos de calidad clave:

* **AQ-01: Disponibilidad:** Se refiere a la capacidad del sistema para estar accesible y operativo para los usuarios la mayor parte del tiempo posible, minimizando las interrupciones. Afecta a usuarios finales, Dirección, IT y Operaciones.
* **AQ-02: Escalabilidad:** Describe la facilidad con la que el sistema puede crecer y soportar un aumento exponencial en el número de usuarios, la cantidad de datos y la complejidad de las relaciones genealógicas. Es de interés para la Dirección, IT, Operaciones y usuarios finales.
* **AQ-03: Rendimiento:** Mide la capacidad del sistema para ofrecer tiempos de respuesta bajos y procesar transacciones eficientemente, incluso bajo alta concurrencia y al realizar consultas complejas. Impacta a usuarios finales, aplicaciones externas, QA e IT.
* **AQ-04: Seguridad:** Implica la protección de los datos contra accesos no autorizados, la garantía de la privacidad de la información y el cumplimiento de normativas relevantes como GDPR y CCPA. Es crucial para usuarios finales, el equipo de Seguridad, Dirección y QA.
* **AQ-05: Interoperabilidad:** Define la facilidad con la que el sistema puede integrarse y comunicarse con aplicaciones de terceros, principalmente a través de APIs públicas y estandarizadas. Relevante para aplicaciones externas, desarrolladores e IT.
* **AQ-06: Usabilidad:** Se centra en la facilidad de uso y la experiencia intuitiva que el sistema proporciona a los usuarios finales en todas las plataformas soportadas (web, móvil). Interesa a usuarios finales, el equipo de UX/UI y QA.
* **AQ-07: Mantenibilidad:** Describe la facilidad para realizar actualizaciones, correcciones de errores y mejoras en el sistema sin afectar negativamente su operación o estabilidad. Es importante para IT, QA y Operaciones.
* **AQ-08: Portabilidad:** Es la capacidad del sistema o de sus componentes para ejecutarse en diferentes entornos y plataformas (web, móvil, diferentes proveedores de nube), conservando su funcionalidad y calidad. Afecta a usuarios finales, IT, QA y el equipo de UX/UI.
* **AQ-09: Integridad de datos:** Garantiza que los datos almacenados, procesados y consultados sean precisos, completos y hayan sido verificados correctamente, lo cual es fundamental para la confianza en la plataforma. De interés para usuarios finales, historiadores y QA.
* **AQ-10: Modificabilidad:** Se refiere a la facilidad con la que el sistema puede ser adaptado o ampliado para responder a cambios en los requisitos de negocio o tecnológicos a lo largo del tiempo. Interesa a la Dirección, IT, QA y usuarios finales.

# Identificación de interesados

Los interesados (stakeholders) son individuos, grupos u organizaciones que afectan o son afectados por el sistema. Su identificación es clave para asegurar que sus necesidades y expectativas sean consideradas en el diseño arquitectónico. Los interesados para "¿Quién es tu papá?" son:

* **I-01: Usuarios finales:** Incluye a usuarios generales, genealogistas aficionados o profesionales, y familias que desean construir y explorar su árbol genealógico. Sus principales intereses son un uso sencillo e intuitivo, la privacidad de sus datos, un buen rendimiento del sistema y la confiabilidad de la información presentada.
* **I-02: Historiadores / Voluntarios:** Son expertos y entusiastas encargados de digitalizar, transcribir y validar documentos históricos como censos y registros parroquiales. Su foco está en la integridad y precisión de los datos, así como en la facilidad para ingresar y validar estos registros históricos.
* **I-03: Desarrolladores externos:** Componen la comunidad de creadores de aplicaciones y servicios que utilizan la API pública del sistema para ofrecer funcionalidades adicionales o integrar datos genealógicos en sus propias plataformas. Les interesa la disponibilidad y estabilidad de la API, así como una documentación clara y facilidades para la integración.
* **I-04: Equipo IT / Operaciones:** Es el personal técnico responsable de la infraestructura tecnológica del sistema. Sus principales preocupaciones son la escalabilidad, eficiencia, facilidad de gestión y el mantenimiento continuo de la infraestructura.
* **I-05: Equipo de Seguridad:** Son los responsables de definir e implementar las políticas de seguridad y privacidad del sistema. Se centran en la protección de datos personales, el cumplimiento normativo (GDPR, CCPA), la auditoría de accesos y cambios.
* **I-06: Equipo QA (Calidad):** Este equipo es responsable de asegurar la calidad del software a través de la identificación temprana de errores, la automatización de pruebas y la promoción de la calidad continua del producto.
* **I-07: Equipo UX/UI:** Son los especialistas en diseño de interfaces de usuario y experiencia de usuario. Su objetivo es garantizar la usabilidad, accesibilidad y una experiencia atractiva e intuitiva para los usuarios en todas las plataformas.
* **I-08: Dirección / Inversionistas:** Comprenden a los responsables de la estrategia empresarial y financiera del proyecto. Sus intereses principales son la rentabilidad, el crecimiento del proyecto, el retorno de la inversión y la reputación de la plataforma.
* **I-09: Product Manager (PM):** Es el encargado de gestionar los requerimientos del producto, asegurando que los objetivos de negocio se alineen con las necesidades reales de los usuarios y priorizando las funcionalidades a desarrollar.

# Interesados VS Requerimientos No Funcionales (Ponderación)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Stakeholder | Funcionalidad | Desempeño | Compatibilidad | Usabilidad | Fiabilidad | Seguridad | Mantenibilidad | Portabilidad | Total |
| Usuarios finales | 20 | 25 | 10 | 25 | 10 | 5 | 3 | 2 | **100** |
| Historiadores/Voluntarios | 20 | 10 | 10 | 20 | 25 | 5 | 5 | 5 | **100** |
| Desarrolladores externos | 25 | 15 | 20 | 5 | 10 | 10 | 5 | 10 | **100** |
| Equipo IT / Operaciones | 10 | 20 | 15 | 5 | 25 | 10 | 10 | 5 | **100** |
| Equipo de Seguridad | 5 | 5 | 5 | 5 | 15 | 60 | 3 | 2 | **100** |
| Total Atributo | **16** | **15** | **12** | **12** | **17** | **18** | **5,2** | **4,8** | **100** |

# Atributos de calidad más importante (Drivers arquitectónicos)

Los drivers arquitectónicos son los atributos de calidad que tienen el mayor impacto en la arquitectura del sistema. Su correcta satisfacción es fundamental para el éxito del proyecto. Para "¿Quién es tu papá?", los siguientes atributos han sido identificados como los drivers arquitectónicos principales:

* **01: Fiabilidad:** Este atributo es crucial porque asegura la integridad y consistencia de los datos genealógicos a lo largo del tiempo. La confianza de los usuarios en la plataforma depende directamente de la fiabilidad de la información que esta provee, especialmente considerando la naturaleza sensible y personal de los datos familiares y la necesidad de preservarlos para futuras generaciones.
* **02: Seguridad:** Dada la naturaleza privada y legalmente protegida de la información personal y genealógica (cumplimiento con GDPR, CCPA), la seguridad es imprescindible. El sistema debe garantizar la confidencialidad, integridad y disponibilidad de los datos frente a accesos no autorizados o brechas de seguridad.
* **03: Eficiencia de Desempeño:** Es esencial garantizar tiempos de respuesta rápidos y un procesamiento eficiente, especialmente ante la expectativa de consultas masivas sobre grafos familiares que pueden llegar a ser extremadamente grandes y complejos. Una buena eficiencia de desempeño impacta directamente la experiencia del usuario y la viabilidad de las aplicaciones de terceros.
* **04: Escalabilidad:** Este atributo es determinante para soportar el crecimiento continuo y potencialmente exponencial del sistema, tanto en términos de millones de usuarios activos como en el volumen de datos, que podría alcanzar miles de millones de nodos y relaciones en el grafo genealógico. La arquitectura debe poder crecer de manera flexible y costo-efectiva.
* **05: Idoneidad Funcional:** Es fundamental que el sistema cumpla correctamente con los requisitos funcionales clave para los cuales fue concebido. Esto incluye la gestión precisa de personas y relaciones, la importación de datos, la visualización del árbol y las capacidades de búsqueda avanzada, que son el núcleo de la propuesta de valor del sistema.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RNF | Descripción | Métrica | | Impacto (1-3) | Dificultad (1-3) | Peso (%) | Valor = (I + D) × Peso |
| Idoneidad funcional | Gestión de personas, relaciones, importación GEDCOM/Facebook, visualización infinita. | 95% funcionalidades clave cubiertas; Tasa de éxito en pruebas. | 3 | | 2 | 16% | 0,8 |
| Eficiencia de desempeño | Consultas rápidas al grafo; soporte de 10,000 RPS en picos. | p95 < 3000ms; pruebas de carga. | 3 | | 3 | 15% | 0,9 |
| Compatibilidad | Integración con APIs/formatos y compatibilidad cross-platform. | Pruebas superadas; importaciones/exportaciones sin errores. | 2 | | 2 | 12% | 0,48 |
| Usabilidad | Flujo en ≤ 3 clics, navegación clara y filtros intuitivos. | Tareas clave ≤ 3 clics; encuesta SUS > X. | 3 | | 2 | 12% | 0,6 |
| Fiabilidad | Recuperación tras fallos; alta disponibilidad; consistencia del grafo. | RPO < 15min; SLA > 99.9%. | 3 | | 3 | 17% | 1,02 |
| Seguridad | Cumplimiento GDPR/CCPA, cifrado, control de acceso, auditoría. | 100% datos cifrados; 0 brechas. | 3 | | 3 | 18% | 1,08 |
| Mantenibilidad | Microservicios, CI/CD, pruebas automatizadas, ingesta de fuentes nuevas. | Cambio menor < X horas; cobertura pruebas > Y%. | 2 | | 2 | 5,20% | 0,208 |
| Portabilidad | SPA/PWA/nativo; migración entre nubes sin retrabajo mayor. | 70% de lógica compartida; PoC migración exitosa. | 2 | | 3 | 4,80% | 0,24 |
|  |  |  |  | |  | 100% | 5,328 |

# Identificación de las tácticas para atacar los atributos de calidad

Para cada driver arquitectónico identificado, se han seleccionado tácticas específicas que ayudarán a alcanzar los niveles deseados de calidad.

**8.1 Tácticas para la Fiabilidad** La fiabilidad se abordará mediante las siguientes tácticas:

* **Backup & Restore automático:** Se realizarán copias de seguridad periódicas y automáticas de las bases de datos y archivos críticos para permitir una recuperación rápida y completa ante incidentes o pérdida de datos.
* **Redundancia de datos:** Se implementará el almacenamiento redundante de la información crítica en múltiples ubicaciones o réplicas para evitar la pérdida de información debido a fallos de hardware o de un único punto de almacenamiento.
* **Mecanismos de reintento:** Las operaciones críticas, como las escrituras en el grafo genealógico, contarán con mecanismos automáticos de reintento para manejar fallos transitorios de red o de servicio.
* **Monitoreo proactivo:** Se establecerá un sistema de monitoreo continuo que genere alertas tempranas ante la detección de anomalías o comportamientos inesperados, permitiendo actuar preventivamente para evitar fallos mayores.
* **Failover activo-activo:** Se utilizará balanceo de carga y replicación multi-zona para asegurar la continuidad operativa, permitiendo que el sistema siga funcionando incluso si una parte de la infraestructura falla.

**8.2 Tácticas para la Seguridad** Para garantizar la seguridad del sistema, se aplicarán las siguientes tácticas:

* **Control de acceso granular (RBAC):** Se implementará una gestión detallada de roles y permisos, permitiendo un control fino sobre qué usuarios pueden acceder o modificar qué información, potencialmente a nivel de nodo o rama familiar.
* **OAuth 2.0 y OIDC:** Se utilizarán estos estándares abiertos para una autenticación robusta, facilitando el inicio de sesión mediante proveedores sociales (Facebook, Google) y la posible implementación de autenticación multifactor (MFA) para mayor seguridad.
* **WAF (Firewall de Aplicación Web):** Se desplegará un WAF para proteger la plataforma contra ataques web comunes como Cross-Site Scripting (XSS), SQL Injection, ataques de Denegación de Servicio Distribuido (DDoS) y abuso de APIs.

**8.3 Tácticas para la Eficiencia de Desempeño** La eficiencia en el desempeño se logrará a través de:

* **Caché distribuida (Redis):** Se implementará un sistema de caché distribuida, como Redis (AWS ElastiCache), para almacenar temporalmente los resultados de consultas frecuentes, como subárboles familiares o perfiles muy consultados, reduciendo la latencia.
* **Búsqueda optimizada (índices):** Se hará un uso extensivo de índices eficientes en las bases de datos de grafos (AWS Neptune) y en otras bases de datos para acelerar las operaciones de búsqueda y consulta.
* **Replicación de lectura:** Se configurarán réplicas de lectura de las bases de datos principales para distribuir la carga de consultas de lectura, mejorando los tiempos de respuesta y reduciendo la carga sobre la instancia principal de escritura.
* **CDN (CloudFront):** Se utilizará una Red de Distribución de Contenido (CDN) como AWS CloudFront para distribuir eficientemente el contenido estático (imágenes, CSS, JavaScript) y los medios digitalizados (escaneos de documentos) desde ubicaciones geográficamente cercanas a los usuarios.

**8.4 Tácticas para la Escalabilidad** Para asegurar la escalabilidad del sistema, se emplearán las siguientes tácticas:

* **Arquitectura en microservicios:** El sistema se dividirá en servicios más pequeños, independientes y desacoplados, cada uno responsable de una funcionalidad de negocio específica. Esto permite que cada servicio pueda crecer y escalar de forma independiente según sus necesidades.
* **Escalado horizontal automático (Auto-scaling):** Se configurará el autoescalado dinámico de los recursos computacionales (contenedores, funciones serverless) para que el sistema pueda ajustarse automáticamente a las variaciones en la demanda, añadiendo o eliminando instancias según sea necesario.
* **Bases de datos NoSQL (grafos):** Se utilizarán bases de datos especializadas, como las bases de datos de grafos (AWS Neptune), que están diseñadas para almacenar y consultar eficientemente grandes volúmenes de datos interrelacionados, superando las limitaciones de las bases de datos relacionales tradicionales para este tipo de estructuras.
* **Balanceo de carga (ALB/NLB):** Se utilizarán balanceadores de carga (Application Load Balancer y/o Network Load Balancer) para distribuir de manera eficiente el tráfico de los usuarios y de las aplicaciones externas entre las múltiples instancias de los microservicios.

**8.5 Tácticas para la Idoneidad Funcional** Para garantizar que el sistema cumple correctamente con sus funciones, se aplicarán estas tácticas:

* **Pruebas de aceptación automatizadas (BDD):** Se implementarán pruebas de aceptación automatizadas, preferiblemente siguiendo un enfoque de Desarrollo Guiado por Comportamiento (BDD), para validar continuamente que el sistema cumple con los criterios de aceptación definidos por los usuarios reales y los stakeholders.
* **Integración continua (CI/CD):** Se establecerá un pipeline de integración y despliegue continuo que automatice las pruebas (unitarias, de integración, de aceptación) y los despliegues a los diferentes entornos, asegurando que cada cambio introduzca funcionalidades correctas y no rompa las existentes.
* **Verificación dual humana (árbitro):** Para los registros históricos, se implementará un proceso donde dos transcripciones de los documentos son realizadas y luego comparadas. En caso de discrepancias, un arbitraje humano intervendrá para asegurar la máxima precisión de los datos ingresados. (Con referencia a RF-06)
* **Auditoría y control de versiones:** El sistema permitirá revisar, auditar y, si es necesario, revertir fácilmente los cambios realizados en la información genealógica, proporcionando un historial de modificaciones. (Con referencia a RF-08)

# Patrones arquitectónicos que a utilizar

* **P-01: Microservicios:** Este es el patrón arquitectónico fundamental del sistema. La aplicación se descompondrá en un conjunto de servicios pequeños, autónomos y cohesivos. Cada microservicio se centrará en una capacidad de negocio específica (ej.: gestión de genealogía, búsquedas especializadas, ingestión de documentos, autenticación, moderación de contenido). La principal justificación es que permite la modularidad, escalabilidad independiente por servicio, despliegues autónomos, diversidad tecnológica (si es necesario por servicio) y resiliencia, ya que un fallo en un servicio no necesariamente detiene todo el sistema.
* **P-02: API Gateway:** Se utilizará un API Gateway como único punto de entrada para todas las solicitudes de los clientes (aplicaciones web, móviles) y de las aplicaciones de terceros. Este patrón proporciona una interfaz unificada y desacoplada, gestionando tareas transversales como la autenticación, autorización, enrutamiento de solicitudes, monitoreo del tráfico, limitación de tasa (rate limiting) y transformación de protocolos. Simplifica la complejidad para los clientes y protege los microservicios subyacentes.
* **P-03: Database per Service (Base de Datos por Servicio):** Cada microservicio tendrá su propia base de datos privada, accesible únicamente a través de la API del microservicio. Esto refuerza el bajo acoplamiento entre servicios y permite que cada uno elija el tipo de base de datos más adecuado para sus necesidades particulares. Por ejemplo, se utilizará una base de datos de grafos (como AWS Neptune) para el servicio de genealogía que maneja relaciones familiares complejas, mientras que se podría usar una base de datos documental (como PostgreSQL o DynamoDB) para la gestión de documentos digitalizados o registros administrativos.
* **P-04: Cache Distribuida (Distributed Cache - ej. Redis):** Se implementará una capa de caché distribuida para mejorar significativamente el rendimiento y la eficiencia de las consultas recurrentes. Almacenará temporalmente los resultados de operaciones costosas o datos accedidos frecuentemente (como subárboles familiares populares o perfiles de usuario) en una memoria rápida como Redis (gestionado por AWS ElastiCache), reduciendo la latencia para el usuario y la carga sobre los servicios y bases de datos principales.
* **P-05: Balanceador de Carga (Load Balancer):** Este patrón es esencial para asegurar la escalabilidad y la alta disponibilidad del sistema. Se utilizarán balanceadores de carga para distribuir eficientemente el tráfico de red entrante entre múltiples instancias de los microservicios. Esto previene la sobrecarga de instancias individuales, mejora los tiempos de respuesta y permite realizar mantenimientos o despliegues sin interrumpir el servicio.
* **P-06: Serverless (FaaS - Functions as a Service):** Se emplearán funciones serverless (ej. AWS Lambda) para la ejecución bajo demanda de tareas específicas y, a menudo, esporádicas o de corta duración. Algunos ejemplos de aplicación incluyen la validación de documentos recién subidos, el envío de notificaciones, el procesamiento de imágenes o tareas de mantenimiento programadas. Este patrón permite optimizar costos (se paga solo por el tiempo de ejecución) y escalar automáticamente sin gestionar servidores.

# Estrategia (Táctica con los Patrones Seleccionados)

La estrategia arquitectónica general consiste en aplicar los patrones y tácticas identificados para satisfacer los drivers arquitectónicos prioritarios. Esta estrategia se detalla a continuación para cada atributo de calidad clave:

**10.1 Estrategia para la Fiabilidad** La fiabilidad se logrará mediante una combinación de redundancia, monitoreo y recuperación rápida.

* **Tácticas Clave Aplicadas:** Backup & Restore automático, Redundancia de datos (a nivel de infraestructura y bases de datos), Mecanismos de reintento en la comunicación entre servicios y con las bases de datos, Monitoreo proactivo para la detección temprana de fallos, y Failover activo-activo entre zonas de disponibilidad o regiones.
* **Patrones Arquitectónicos Asociados:** La arquitectura de **Microservicios** contribuye al aislamiento de fallos; un problema en un servicio no debería derribar todo el sistema. Las funciones **Serverless (FaaS)** pueden utilizarse para tareas de recuperación o procesamiento asíncrono resiliente. El **Balanceador de Carga** es fundamental para el failover y la distribución de carga que previene la sobrecarga de instancias.
* **Descripción de la Estrategia:** Se implementarán microservicios independientes, cada uno con sus propias bases de datos replicadas (utilizando el patrón Database per Service) y con políticas de backup automáticas para prevenir la pérdida de datos. El balanceador de carga, en conjunto con el despliegue en múltiples zonas de disponibilidad (y potencialmente regiones), asegurará la alta disponibilidad mediante mecanismos de failover. Las funciones serverless permitirán la ejecución escalable y resiliente de funciones críticas de sistema o recuperación. El monitoreo constante alertará sobre cualquier degradación para una acción correctiva rápida.

**10.2 Estrategia para la Seguridad** La seguridad se integrará en todas las capas del sistema, desde el perímetro hasta el nivel de datos y aplicación.

* **Tácticas Clave Aplicadas:** Control de acceso granular (RBAC) para definir permisos detallados, uso de OAuth 2.0 y OIDC para autenticación robusta, Cifrado en tránsito (TLS) y en reposo (AES-256) para proteger la confidencialidad de los datos, y un WAF (Firewall de Aplicación Web) para la protección contra ataques comunes.
* **Patrones Arquitectónicos Asociados:** El **API Gateway** actuará como el principal punto de control para la seguridad, centralizando la autenticación y autorización de todas las solicitudes entrantes mediante OAuth 2.0 / OIDC y aplicando políticas de RBAC. La arquitectura de **Microservicios** permite aplicar contextos de seguridad específicos a cada servicio si es necesario.
* **Descripción de la Estrategia:** El API Gateway será la primera línea de defensa, gestionando la autenticación y autorización de todos los clientes y aplicaciones externas. Dentro de la plataforma, el WAF protegerá contra amenazas a nivel de aplicación. La comunicación entre servicios y el almacenamiento de datos se protegerán con cifrado. Se implementará un control de acceso granular para asegurar que los usuarios solo puedan acceder a la información que les corresponde.

**10.3 Estrategia para la Eficiencia de Desempeño** Se buscará optimizar los tiempos de respuesta y la capacidad de procesamiento mediante múltiples capas de mejora.

* **Tácticas Clave Aplicadas:** Uso extensivo de Caché distribuida (Redis) para datos de acceso frecuente, Búsqueda optimizada mediante índices en las bases de datos (especialmente en la base de datos de grafos), Replicación de lectura para descargar las bases de datos principales, uso de CDN (CloudFront) para contenido estático, y la consideración de Sharding por familias (subárbol) para grafos muy grandes.
* **Patrones Arquitectónicos Asociados:** El patrón de **Cache Distribuida (Redis)** es central aquí. La arquitectura de **Microservicios** permite optimizar el rendimiento de servicios individuales. El patrón **Database per Service** permite elegir la base de datos óptima para el rendimiento de cada servicio. El **Balanceador de Carga** distribuye la carga para mantener el rendimiento. El uso de una **CDN** es un patrón en sí mismo para la distribución de contenido.
* **Descripción de la Estrategia:** Se utilizará Redis para cachear consultas y datos populares, reduciendo drásticamente la latencia para los usuarios. Las bases de datos especializadas (Neptune para grafos) se optimizarán con índices y, si es necesario, técnicas de sharding para consultas eficientes. Los balanceadores de carga distribuirán el tráfico, y la CDN acercará el contenido estático a los usuarios, minimizando los tiempos de carga.

**10.4 Estrategia para la Escalabilidad** La arquitectura se diseñará para crecer horizontalmente y adaptarse a la demanda fluctuante.

* **Tácticas Clave Aplicadas:** Adopción de una Arquitectura en microservicios para el escalado independiente de componentes, Escalado horizontal automático de los recursos computacionales, uso de Bases de datos NoSQL (especialmente de grafos) diseñadas para grandes volúmenes, y Balanceo de carga para distribuir el tráfico entre las instancias escaladas.
* **Patrones Arquitectónicos Asociados:** La arquitectura de **Microservicios** es fundamental, ya que permite que cada servicio escale de forma independiente. El patrón **Database per Service** soporta esta escalabilidad al permitir que las bases de datos también escalen según las necesidades del servicio. El **Balanceador de Carga** es crucial para distribuir el tráfico a las instancias escaladas. Las funciones **Serverless (FaaS)** ofrecen escalabilidad inherente para tareas específicas.
* **Descripción de la Estrategia:** La naturaleza desacoplada de los microservicios permitirá escalar cada funcionalidad del sistema de forma independiente, optimizando el uso de recursos. Se configurará el escalado automático para los servicios y las funciones serverless, permitiendo al sistema adaptarse dinámicamente a los picos de demanda. Las bases de datos NoSQL especializadas en grafos están diseñadas para manejar volúmenes masivos de datos interconectados y escalar eficientemente.

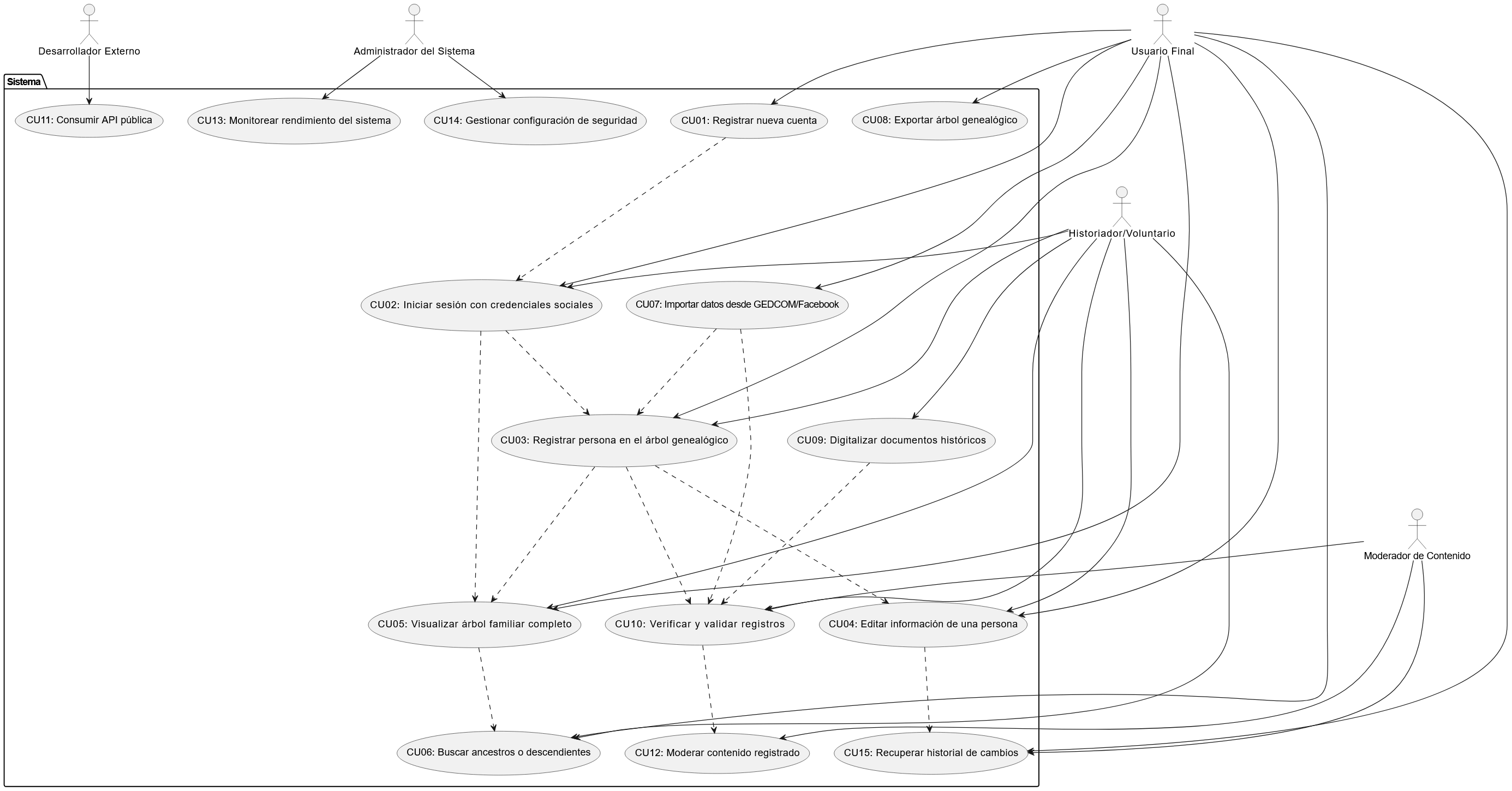
**10.5 Estrategia para la Idoneidad Funcional** Se asegurará que el sistema cumpla con los requisitos funcionales mediante un proceso de desarrollo y validación riguroso.

* **Tácticas Clave Aplicadas:** Implementación de Pruebas de aceptación automatizadas (BDD) para validar el comportamiento esperado, un pipeline de Integración continua (CI/CD) para asegurar la calidad en cada cambio, el proceso de Verificación dual humana para la precisión de los datos históricos, mecanismos de Auditoría y control de versiones para la trazabilidad y reversibilidad de cambios, y el uso de Feature toggles para el despliegue controlado de funcionalidades.
* **Patrones Arquitectónicos Asociados:** La arquitectura de **Microservicios** facilita las pruebas y despliegues independientes, contribuyendo a la calidad funcional al aislar los cambios. Las funciones **Serverless (FaaS)** pueden usarse para tareas de validación o procesamiento específicas dentro de los flujos funcionales. El **API Gateway** asegura que las interfaces funcionales se expongan de manera consistente.
* **Descripción de la Estrategia:** Se establecerán pipelines de CI/CD que incluyan múltiples capas de pruebas automatizadas (unitarias, de integración, de aceptación BDD) para validar que cada funcionalidad se implementa correctamente y sigue cumpliendo los criterios definidos. El proceso de verificación dual humana para los registros históricos es una táctica clave para la calidad de los datos. La auditoría y el control de versiones permitirán rastrear y, si es necesario, revertir cambios, mientras que los feature toggles facilitarán el despliegue gradual y seguro de nuevas funcionalidades, minimizando el riesgo de regresiones.

# Modelo 4 + 1

* Vista Lógica: La Vista Lógica describe la estructura del sistema en términos de sus principales abstracciones y elementos funcionales, sus responsabilidades y las relaciones entre ellos. Se enfoca en satisfacer los requerimientos funcionales y representa la perspectiva del diseñador sobre cómo el sistema está organizado conceptualmente.
* Vista de Implementación: "La Vista de Implementación describe cómo se organiza el software en módulos, componentes y paquetes en el entorno de desarrollo. Detalla las dependencias entre estos elementos y cómo se mapean a los artefactos desplegables. Esta vista es crucial para la gestión de la configuración, la integración y las pruebas."
* Vista de Procesos: "La Vista de Procesos se enfoca en los aspectos dinámicos del sistema, ilustrando cómo los diferentes componentes interactúan en tiempo de ejecución para realizar las funcionalidades clave. Se centra en la concurrencia, la sincronización, el rendimiento y la escalabilidad de los procesos."
* Vista Física (o de Despliegue): "La Vista Física describe la topología del hardware y software en la que se despliega el sistema. Muestra cómo los componentes de software se mapean a los nodos físicos (servidores, contenedores, etc.) y cómo estos nodos se comunican entre sí. Es fundamental para entender la escalabilidad, la fiabilidad y la disponibilidad del sistema."

## 11.1 Escenarios

<https://drive.google.com/file/d/1H4dvUsvIQZZ_OMLtsT4ZE6xMG0vSMXo9/view?usp=drive_link>

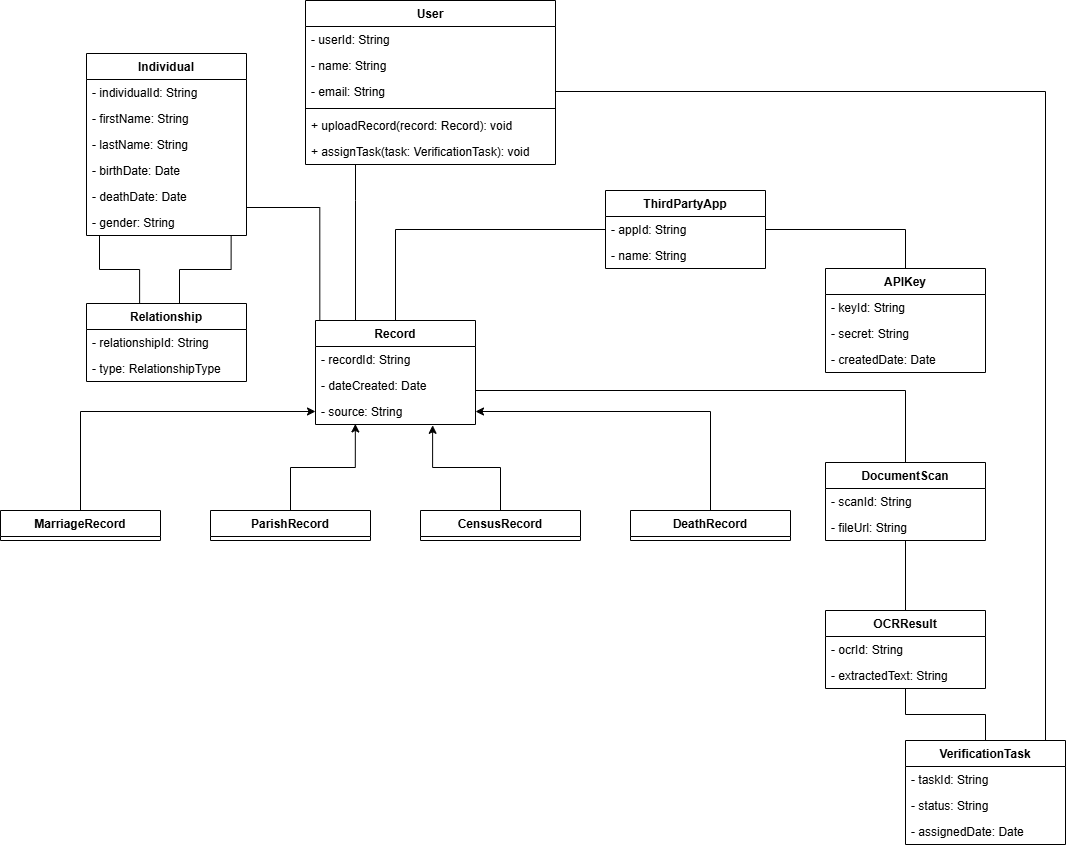
Actores

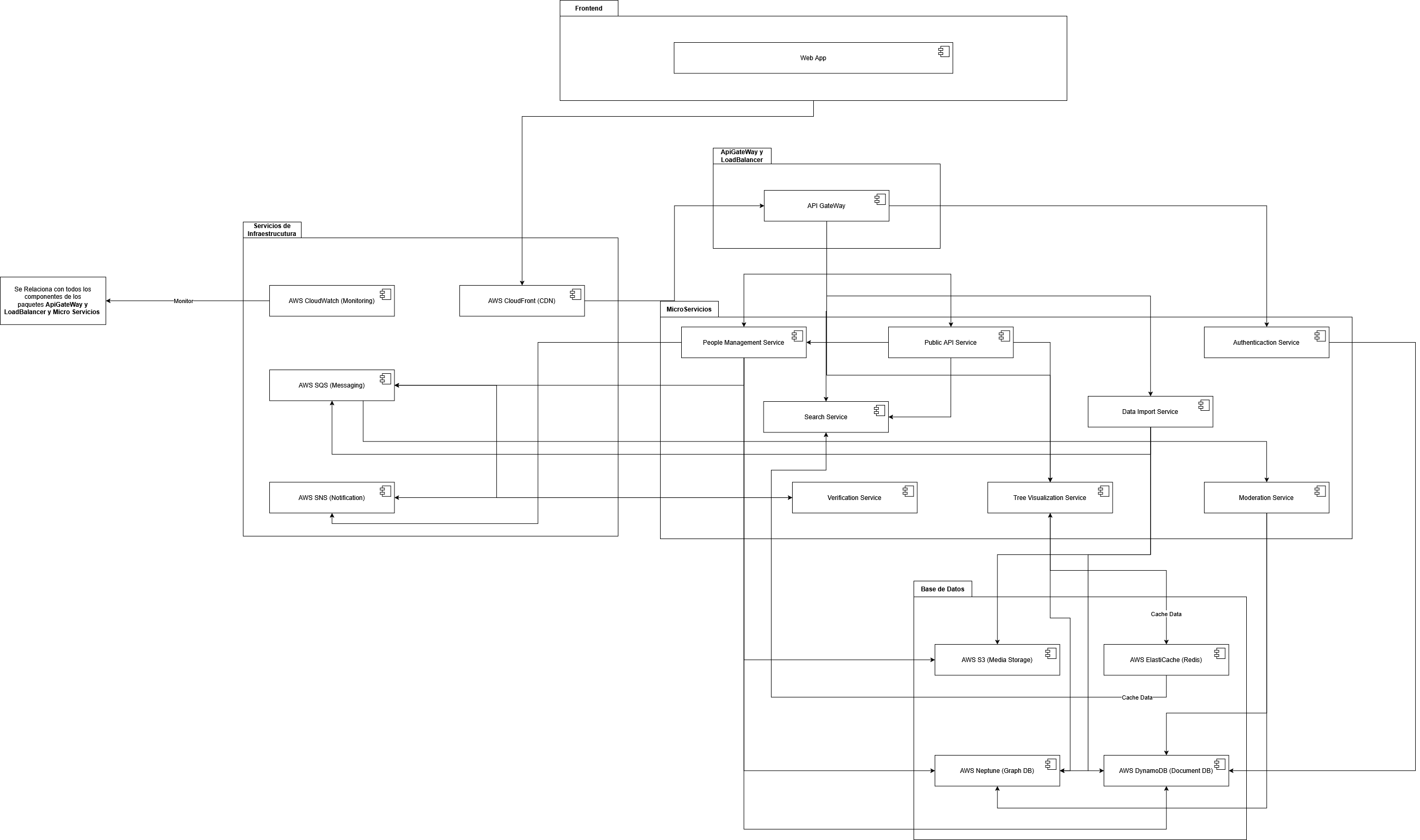
* **Usuario Final**: Personas que buscan construir su árbol genealógico.
* **Historiador/Voluntario**: Expertos que digitalizan y validan documentos históricos.
* **Desarrollador Externo**: Programadores que consumen la API pública.
* **Administrador del Sistema**: Personal técnico que gestiona la plataforma.
* **Moderador de Contenido**: Validadores que verifican la precisión de los datos.

### Casos de Uso

* **CU01: Registrar nueva cuenta**: Proceso de creación de cuenta de usuario.
* **CU02: Iniciar sesión con credenciales sociales**: Login mediante Facebook/Google.
* **CU03: Registrar persona en árbol genealógico**: Añadir nuevo miembro familiar.
* **CU04: Editar información de una persona**: Modificar datos de un miembro existente.
* **CU05: Visualizar árbol familiar completo**: Ver el árbol con zoom y navegación.
* **CU06: Buscar ancestros o descendientes**: Búsquedas avanzadas por relación.
* **CU07: Importar datos desde GEDCOM/Facebook**: Ingesta de datos externos.
* **CU08: Exportar árbol genealógico**: Generación de archivos para uso externo.
* **CU09: Digitalizar documentos históricos**: Conversión de documentos físicos.
* **CU10: Verificar y validar registros**: Proceso de verificación dual.
* **CU11: Consumir API pública**: Uso de endpoints por desarrolladores externos.
* **CU12: Moderar contenido registrado**: Revisión y aprobación de contenido.
* **CU13: Monitorear rendimiento del sistema**: Supervisión técnica.
* **CU14: Gestionar configuración de seguridad**: Administración de políticas.
* **CU15: Recuperar historial de cambios**: Ver versiones anteriores del árbol.

## 11.2 Vista Lógica



11.3 Vista de Implementación<https://drive.google.com/file/d/1U-xWJrAiu9kgxTIn6HsCGSBMR5AuTwdi/view?usp=drive_link>

Frontend

* **Web App** : Aplicación frontend para navegadores web.

### API Gateway & Load Balancer

* **API Gateway**: Punto de entrada centralizado para todas las API del sistema.

### Servicios de Microservicios

* **Authentication Service**: Gestiona autenticación y autorización.
* **People Management Service**: Administra nodos (personas) y sus relaciones.
* **Search Service**: Implementa algoritmos de búsqueda especializada en grafos.
* **Tree Visualization Service**: Genera visualizaciones renderizables del árbol.
* **Data Import Service**: Procesa y valida datos de fuentes externas.
* **Verification Service**: Implementa el sistema de arbitraje dual para registros.
* **Moderation Service**: Gestiona revisiones y administra cambios.
* **Public API Service**: Expone endpoints para desarrolladores externos.

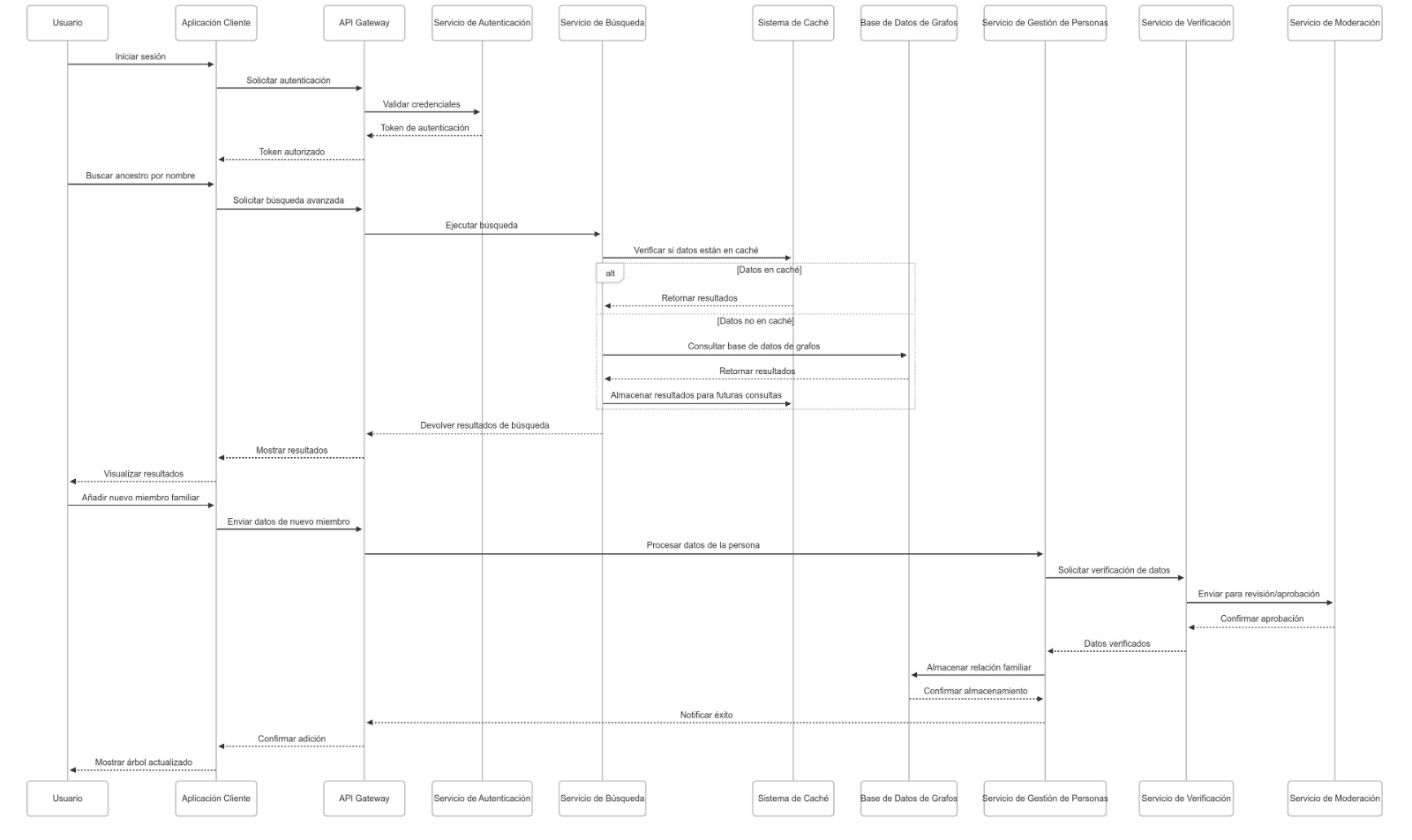
### Almacenamiento de Datos

* **AWS Neptune**: Base de datos de grafos especializada para relaciones complejas.
* **AWS DynamoDB**: Base de datos NoSQL para documentos históricos.
* **AWS S3**: Almacenamiento de objetos para medios digitales.
* **AWS ElastiCache**: Sistema de caché distribuido (Redis) para consultas frecuentes.

### Servicios de Infraestructura

* **AWS CloudFront**: CDN para distribución global de contenido estático.
* **AWS SQS**: Cola de mensajes para comunicación asíncrona entre servicios.
* **AWS SNS**: Sistema de notificaciones para eventos del sistema.
* **AWS CloudWatch**: Monitoreo y alertas para todos los componentes.

## 11.4 Vista de Procesos

<https://mermaid.live/edit#pako:eNqNlUtz2jAQx7-KRmdIY0gI-JAZXiUc0jJJyKHDZbEXosaWqCzTPCYfJsccO731yhfrSraJwaRTX7DRb1f7-K_0zAMVIvd5gj9SlAEOBCw1xDPJ6FmBNiIQK5CGTRkkbJqkoIWqrnYnE7veXUUigEBsfkvWjwRKg4fYsWPpZwQGf8LjAWZ6c3FtqWvUa_pTsRBZNzXksdiganR91b_Yt-lt_iSUWQhVvN_tXwwdLxKDMVi8D8Hd5q3KjgY9S_YgQYsNwKjEvow0LFRS5SfDq0r4I0yMqwy9T1AnSsIBy9vh1fjzvuktarH4OPHLr4N9i0tqqy74zGJaPz-nPvlsLIkCzRJM3v3Rilsf--xaUReFIQIOVJwQC9oG-ewWIhESGGgMST4CIsxzckC9cHmj7tEl_pHHIrYMJEpp8QShqgTfS5PARkZaTYxWbKU0kyqea_xHHvNCBQzWIMkvlHOxuvHZ8DsG6Q6cMXaVIKcWf9sJqp5goZMBxbF5lYzCDkrqgcjkMtlbsI9zVt9ufYWUriSfGpM0MpR2XkSMkkJsUh1ylAdH8vRZX0lnTRnkMg0LmS5LMrUP8f-xezX7bhRDgLusVSKwRWpSTRoM8iCKBGRYqmLRmAGuVbTGHS8U5V7hy7K4VNTsAyHabhMzpcYIOpsi0swutKee7uYXhIIUk1IILBZIwlFsAbGIaCIqChrKtR2UbR13zMoSsvPus4lWpMqyQQRslY16RluOcDfkZYGuywO-bV1m42AyoinfRuRqrnEt3AR_gpVW8_JIEVt_34eUsRA6tmOzD2bOtxlkWiui2ZY5D9vJrKyBKA94t4C5ujKPpb1zw9jeC6rkuKj2F2Xy2dq8PQijqjIoOQtFsHt4ZTIohLJ51XMVMQhMpopQ8RpfahFy3-gUazxG8mM_-bP1MePmDmOccZ9e6US7n_GZfCEbOmC_KRUXZlqlyzvuL4Ams8bTFfWpuDO3CIkedV-l0nC_2Wo7H9x_5g_c9xreUeu03fA6J16nceadNGr8kaizo-PTlnfc9M46ba_d7LzU-JPb1Tvymh3vrNloWbx9etyqcQypOPoyu7jd_f3yF9Lulxs>

### Proceso de Autenticación

* **Usuario → Aplicación Cliente**: El usuario inicia el proceso de login.
* **Aplicación → API Gateway → Autenticación**: La solicitud pasa por gateway para su enrutamiento.
* **Autenticación → API → Aplicación**: Se validan credenciales y se devuelve un token de acceso.

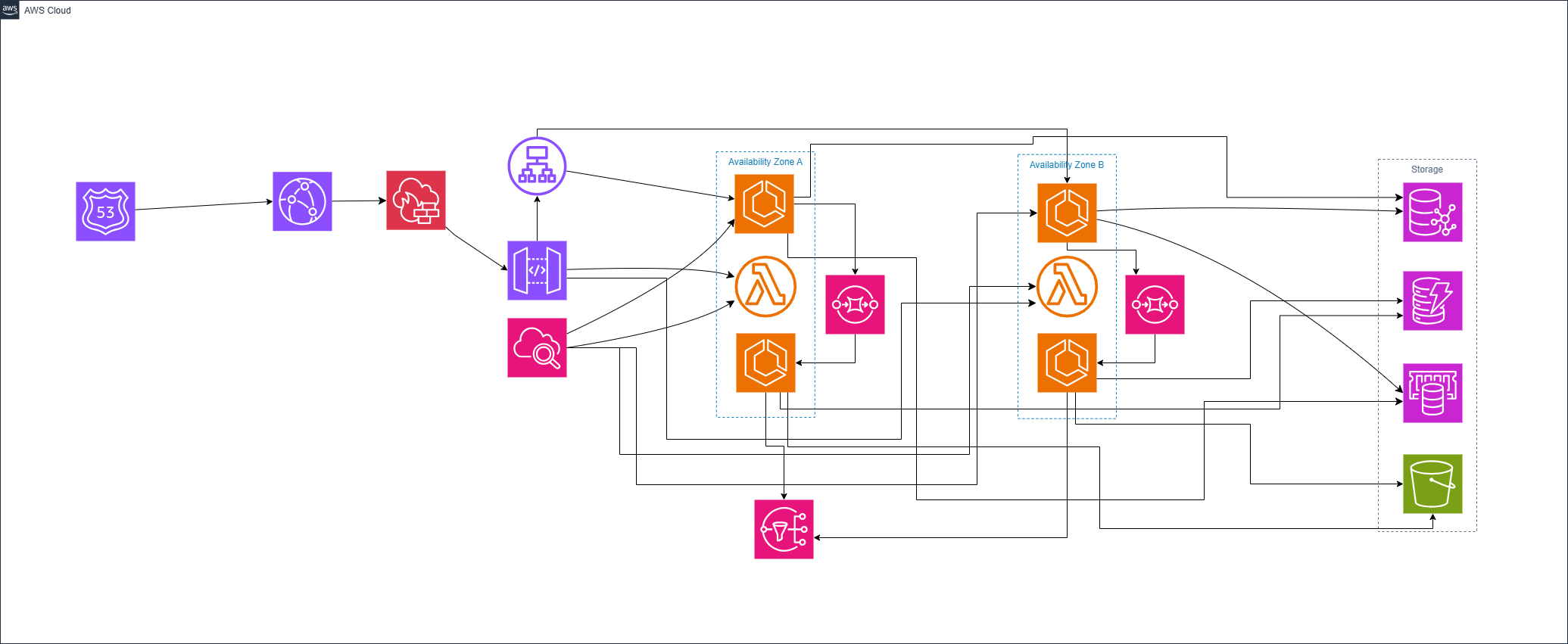
### Proceso de Búsqueda

* **Usuario → Aplicación → API → Búsqueda**: El usuario solicita buscar un ancestro.
* **Búsqueda → Caché**: El sistema verifica si la consulta está en caché para respuesta rápida.
* **Caché o DB → Búsqueda → API → App → Usuario**: Se retornan resultados (desde caché o base de datos).

### Proceso de Adición de Familiar

* **Usuario → App → API → Gestión de Personas**: El usuario añade un nuevo miembro familiar.
* **Gestión → Verificación → Moderación**: Se inicia el proceso de verificación dual.
* **Moderación → Verificación → Gestión**: Se confirma la aprobación del nuevo miembro.
* **Gestión → Base de Datos → Gestión → API → App → Usuario**: Se almacena y se confirma la nueva relación.

## 11.5 Vista Física



### Por región:

* **Zonas de Disponibilidad (A y B)**: Infraestructura redundante en diferentes centros de datos.
* **ECS Clusters**: Contenedores Docker gestionados para microservicios.
* **Lambda Functions**: Funciones serverless para operaciones esporádicas.

### Capa de Red y Seguridad

* **CloudFront**: CDN global que acelera la entrega de contenido.
* **WAF**: Firewall de aplicaciones web que protege contra ataques comunes.
* **Application Load Balancer**: Distribuye tráfico entre instancias.
* **API Gateway**: Gestiona, autentica y enruta llamadas API.
* **Route 53**: Servicio DNS.

### Almacenamiento

* **Neptune**: Base de datos de grafos con replicación.
* **DynamoDB**: Tablas globales con replicación automática.
* **S3**: Almacenamiento de objetos con replicación cross-region.
* **ElastiCache (Redis)**: Caché en memoria para consultas frecuentes.

### Mensajería y Monitoreo

* **SQS**: Colas de mensajes para procesamiento desacoplado.
* **CloudWatch**: Monitoreo centralizado para todos los recursos.
* **SNS**: Servicio de notificaciones para publicar mensajes a múltiples destinos (email, SMS, etc.) tras eventos clave del sistema.

## 11.6 Relación con Atributos de Calidad Prioritarios

### 1. Fiabilidad

* **Vista Lógica**: Separación de responsabilidades minimiza impacto de fallos.
* **Vista de Procesos**: Verificación dual garantiza precisión de datos.
* **Vista de Componentes**: Servicios desacoplados evitan fallos en cascada.
* **Vista Física**: Múltiples zonas de disponibilidad y regiones con replicación.

### 2. Seguridad

* **Vista Lógica**: Servicio de autenticación centralizado.
* **Vista de Procesos**: Autenticación al inicio de cada proceso.
* **Vista de Componentes**: WAF y servicios de seguridad AWS.
* **Vista Física**: Separación en capas con perímetros de seguridad.

### 3. Eficiencia de Desempeño

* **Vista Lógica**: Caché para consultas frecuentes.
* **Vista de Procesos**: Verificación de caché antes de consultar base de datos.
* **Vista de Componentes**: ElastiCache y CloudFront para aceleración.
* **Vista Física**: Balanceadores de carga y distribución geográfica.

### 4. Escalabilidad

* **Vista Lógica**: Servicios independientes pueden escalar por separado.
* **Vista de Componentes**: Arquitectura de microservicios.
* **Vista Física**: Auto-scaling en ECS y servicios elásticos de AWS.

### 5. Idoneidad Funcional

* **Vista Lógica**: Servicios especializados para cada funcionalidad clave.
* **Vista de Procesos**: Flujos completos para búsqueda y verificación.
* **Vista de Casos de Uso**: Alineados directamente con requerimientos funcionales.