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

Índice

|  |
| --- |
| [1. Contexto 2](#_Toc1605533215)  [2. Requerimientos funcionales (RF) 3](#_Toc1197247968)  [3. Requerimientos No funcionales (RNF) 4](#_Toc1110326682)  [4. Atributos de calidad 5](#_Toc1075294998)  [5. Identificación de interesados 6](#_Toc1675226865)  [6. Interesados VS Requerimientos No Funcionales (Ponderación) 7](#_Toc359491597)  [7. Atributos de calidad más importante (Drivers arquitectónicos) 7](#_Toc535937259)  [8. Identificación de las tácticas para atacar los atributos de calidad 8](#_Toc1751040104)  [9. Los Patrones arquitectónicos que van a utilizar 10](#_Toc1552494794)  [10. Estrategia (Táctica con los Patrones Seleccionados) 10](#_Toc1835867654)  [11. Modelo 4 + 1 12](#_Toc1150945823)  [11.1 Escenarios 13](#_Toc1163388060)  [Casos de Uso 13](#_Toc201159338)  [11.2 Vista Lógica 14](#_Toc1607047541)  [11.3 Vista de Implementación 14](#_Toc86303653)  [API Gateway & Load Balancer 15](#_Toc1324897099)  [Servicios de Microservicios 15](#_Toc998387903)  [Almacenamiento de Datos 15](#_Toc539164837)  [Servicios de Infraestructura 16](#_Toc1830169093)  [11.4 Vista de Procesos 16](#_Toc350852265)  [Proceso de Autenticación 17](#_Toc1577534501)  [Proceso de Búsqueda 17](#_Toc2109712179)  [Proceso de Adición de Familiar 17](#_Toc1623073639)  [11.5 Vista Física 17](#_Toc584305302)  [Multi-Región 18](#_Toc2046211649)  [Por cada región: 18](#_Toc497662176)  [Capa de Red y Seguridad 18](#_Toc494582165)  [Almacenamiento 18](#_Toc1718732548)  [Mensajería y Monitoreo 18](#_Toc1949230894)  [11.6 Relación con Atributos de Calidad Prioritarios 19](#_Toc1005611168)  [1. Fiabilidad 19](#_Toc371393669)  [2. Seguridad 19](#_Toc1112944276)  [3. Eficiencia de Desempeño 19](#_Toc1082603662)  [4. Escalabilidad 19](#_Toc1155495503)  [5. Idoneidad Funcional 19](#_Toc1497299016) |

# Contexto

Este documento presenta la solución arquitectónica propuesta para el desarrollo de **Who’s Your Daddy?**, un sistema innovador diseñado para construir y gestionar el árbol genealógico más extenso y detallado jamás creado. Este proyecto responde a la necesidad creciente de millones de usuarios por descubrir, preservar y compartir información sobre su historia familiar, facilitando el acceso universal a través de aplicaciones web y móviles.

La plataforma permite integrar diversas fuentes históricas digitalizadas como censos, certificados civiles y religiosos, los cuales son cuidadosamente verificados para garantizar la precisión y calidad de los datos. Además, promueve la interoperabilidad mediante una API abierta, lo que permite a terceros desarrollar nuevas aplicaciones y servicios sobre la información disponible.

Debido a la naturaleza dinámica y exponencialmente creciente de los datos genealógicos, la solución planteada enfatiza atributos críticos como la disponibilidad, escalabilidad, rendimiento, seguridad e interoperabilidad. Esta propuesta detalla minuciosamente la estrategia, tácticas, patrones arquitectónicos y vistas técnicas que soportarán eficazmente los objetivos del proyecto.

# Requerimientos funcionales (RF)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | NOMBRE | DETALLE | INTERESADOS |  |
| RF-01 | Gestión de personas | El sistema permite registrar, editar y eliminar personas y sus relaciones familiares (padre, madre, hijos, cónyuge, etc.). | Usuarios finales, Historiadores, Terceros (Apps externas), PM |  |
| RF-02 | Importación de datos externos | El sistema permite importar datos desde plataformas externas como Facebook, GEDCOM, archivos CSV y APIs públicas de registros históricos. | Usuarios finales, Historiadores, IT |  |
| RF-03 | Búsqueda avanzada | El sistema ofrece búsqueda avanzada y global por nombres, apellidos, fechas de nacimiento/muerte y tipo de parentesco, incluyendo autocompletado inteligente. | Usuarios finales, Apps externas, Historiadores |  |
| RF-04 | Visualización del árbol | El sistema ofrece visualización interactiva del árbol genealógico con zoom infinito, filtros por ramas familiares y navegación sencilla en plataformas web y móvil. | Usuarios finales, IT, UX/UI |  |
| RF-05 | API abierto | El sistema proporciona una API pública (REST) accesible para terceros, con capacidades completas de lectura y escritura sobre la información genealógica. | Terceros (Apps externas), IT, Seguridad, Desarrolladores externos |  |
| RF-06 | Verificación dual de registros | Los registros históricos ingresados (censos, certificados, etc.) deben ser digitalizados y validados mediante arbitraje automático. | Historiadores, Seguridad, QA, PM |  |
| RF-07 | Gestión de autenticación | El sistema implementa autenticación y gestión de cuentas de usuario mediante integración social (Facebook, Google) y credenciales propias (email). | Usuarios finales, IT, Seguridad |  |
| RF-08 | Moderación y control de versiones | El sistema proporciona herramientas de moderación, revisión de cambios históricos y posibilidad de deshacer acciones sobre la información del árbol genealógico. | Seguridad, Usuarios finales, Historiadores, PM |  |

# Requerimientos No funcionales (RNF)

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NOMBRE | DETALLE | INTERESADOS |
| RNF-01 | Idoneidad funcional | Cubrir correctamente gestión de personas y relaciones en grafo: registro/edición/búsqueda por parentesco, importación GEDCOM/Facebook, visualización infinita. | Usuarios finales, Historiadores, Dev. ext. |
| RNF-02 | Eficiencia de desempeño | Consultas al grafo (p. ej. “todos los bisabuelos”) en < 300 ms (p95) y capacidad de procesar 10 000 RPS en picos de demandas simultáneas. | Usuarios finales, IT/Ops |
| RNF-03 | Compatibilidad | Integración fluida con Facebook API, GEDCOM, CSV, y navegadores/plataformas (web, PWA, iOS, Android) sin conflictos ni pérdida de datos. | Dev. ext., IT/Ops |
| RNF-04 | Usabilidad | Flujo de usuario para agregar o buscar un familiar en ≤ 3 clics; interfaz clara para ver ramas, zoom infinito y filtros intuitivos en web y móvil. | Usuarios finales, UX/UI |
| RNF-05 | Fiabilidad | Mantener consistencia del grafo tras caídas o fallos parciales: recuperación automática, sin corrupción de datos; tolerancia a desconexiones de red durante ediciones. | IT/Ops, Historiadores, QA |
| RNF-06 | Seguridad | Protección de datos genealógicos (GDPR/CCPA), cifrado TLS/at-rest, control de acceso por niveles (público/privado) y audit trail de cambios críticos. | Seguridad, Usuarios finales |
| RNF-07 | Mantenibilidad | Arquitectura modular en microservicios; CI/CD con tests automáticos para ingestion de nuevos tipos de fuentes históricas y corrección rápida de errores. | IT/Ops, QA, PM |
| RNF-08 | Portabilidad | Clientes (SPA, PWA, iOS, Android) que compartan lógica, y backend migrable entre nubes (AWS/Azure/GCP) sin retrabajo significativo. | IT/Ops, Dev. ext. |

# Atributos de calidad

|  |  |  |  |
| --- | --- | --- | --- |
| ID | ATRIBUTO DE CALIDAD | DESCRIPCIÓN | INTERESADOS |
| AQ-01 | Disponibilidad | Capacidad del sistema para estar accesible y operativo para los usuarios la mayor parte del tiempo. | Usuarios finales, Dirección, IT, Ops |
| AQ-02 | Escalabilidad | Facilidad del sistema para crecer y soportar un aumento exponencial de usuarios, datos y relaciones genealógicas. | Dirección, IT, Ops, Usuarios finales |
| AQ-03 | Rendimiento | Capacidad para ofrecer tiempos de respuesta bajos, incluso en consultas complejas y bajo alta concurrencia. | Usuarios finales, Apps externas, QA, IT |
| AQ-04 | Seguridad | Protección de los datos, control de accesos, cumplimiento normativo (GDPR, CCPA) y privacidad de la información. | Usuarios finales, Seguridad, Dirección, QA |
| AQ-05 | Interoperabilidad | Facilidad para integrarse y comunicarse con aplicaciones de terceros mediante APIs públicas y estandarizadas. | Apps externas, Desarrolladores, IT |
| AQ-06 | Usabilidad | Facilidad de uso y experiencia intuitiva para los usuarios finales en todas las plataformas soportadas. | Usuarios finales, UX/UI, QA |
| AQ-07 | Mantenibilidad | Facilidad para realizar actualizaciones, corrección de errores y mejoras en el sistema sin afectar la operación. | IT, QA, Ops |
| AQ-08 | Portabilidad | Capacidad para ejecutarse en diferentes entornos y plataformas (web, móvil), conservando funcionalidad y calidad. | Usuarios finales, IT, QA, UX/UI |
| AQ-09 | Integridad de datos | Garantía de que los datos almacenados y consultados sean precisos, completos y verificados correctamente. | Usuarios finales, Historiadores, QA |
| AQ-10 | Modificabilidad | Facilidad para adaptar o ampliar el sistema según cambien los requisitos de negocio o tecnológicos. | Dirección, IT, QA, Usuarios finales |

# Identificación de interesados

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Interesado | Rol / Perfil | Responsabilidad e Intereses Principales |
| I-01 | Usuarios finales | Usuarios generales, genealogistas, familias | Uso sencillo e intuitivo, privacidad, rendimiento y confiabilidad de datos. |
| I-02 | Historiadores / Voluntarios | Expertos encargados de digitalizar y validar documentos históricos | Integridad y precisión de los datos, facilidad para ingresar y validar registros históricos. |
| I-03 | Desarrolladores externos | Creadores de aplicaciones que utilizan la API pública | Disponibilidad, estabilidad y claridad en documentación e integración API. |
| I-04 | Equipo IT / Operaciones | Personal técnico encargado de la infraestructura tecnológica | Escalabilidad, eficiencia, facilidad de gestión y mantenimiento de infraestructura. |
| I-05 | Equipo de Seguridad | Responsables de políticas de seguridad y privacidad | Protección de datos personales, cumplimiento normativo (GDPR, CCPA), auditoría y accesos. |
| I-06 | Equipo QA (Calidad) | Responsables de la calidad del software | Identificación temprana de errores, automatización de pruebas y calidad continua del producto. |
| I-07 | Equipo UX/UI | Especialistas en diseño e interfaces de usuario | Usabilidad, accesibilidad y experiencia del usuario atractiva e intuitiva. |
| I-08 | Dirección / Inversionistas | Responsables de estrategia empresarial y financiera | Rentabilidad, crecimiento del proyecto, retorno de inversión y reputación. |
| I-09 | Product Manager (PM) | Encargados de gestionar los requerimientos del producto | Alinear objetivos de negocio con necesidades reales de los usuarios, priorizar funcionalidades. |

# 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)

|  |  |  |
| --- | --- | --- |
| ID | Atributo de Calidad (Driver) | Justificación |
| 01 | **Fiabilidad** | Asegura integridad y consistencia de datos, crucial para la confianza en la información genealógica. |
| 02 | **Seguridad** | Imprescindible debido a la privacidad y protección legal de la información sensible. |
| 03 | **Eficiencia de desempeño** | Garantiza respuestas rápidas, esencial ante consultas masivas sobre grandes grafos familiares. |
| 04 | **Escalabilidad** | Determinante para soportar el crecimiento continuo y exponencial del sistema (millones de usuarios, billones de nodos). |
| 05 | **Idoneidad funcional** | Fundamental para cumplir correctamente con los requisitos funcionales clave del sistema. |

# Identificación de las tácticas para atacar los atributos de calidad

Fiabilidad

|  |  |
| --- | --- |
| Táctica | Descripción |
| Backup & Restore automático | Copias de seguridad periódicas y automáticas para recuperación rápida ante incidentes. |
| Redundancia de datos | Almacenamiento redundante para evitar pérdida de información. |
| Mecanismos de reintento | Reintentos automáticos de operaciones fallidas (escrituras en el grafo). |
| Monitoreo proactivo | Alertas tempranas ante anomalías para actuar preventivamente. |
| Failover activo-activo | Balanceo de carga y replicación multi-zona para continuidad operativa. |

Seguridad

|  |  |
| --- | --- |
| Táctica | Descripción |
| Control de acceso granular (RBAC) | Gestión detallada de roles y permisos por usuario y nodo. |
| OAuth 2.0 y OIDC | Autenticación robusta utilizando estándares abiertos (login social, multifactor opcional). |
| Cifrado en tránsito y reposo | Protección mediante cifrado (TLS, AES-256) para comunicaciones y almacenamiento de información. |
| WAF (Firewall aplicación web) | Protección frente a ataques frecuentes (XSS, SQL Injection, DDoS, API abuse). |

Eficiencia de desempeño

|  |  |
| --- | --- |
| Táctica | Descripción |
| Caché distribuida (Redis) | Cacheo de consultas frecuentes (subárboles familiares más consultados). |
| Búsqueda optimizada (índices) | Uso de índices eficientes en bases de datos de grafos (Neptune, Neo4j). |
| Replicación de lectura | Lectura distribuida entre réplicas de bases de datos para reducir latencia. |
| CDN (CloudFront) | Distribución eficiente de contenido estático y medios digitalizados. |
| Sharding por familias (subárbol) | División lógica del grafo para acelerar consultas locales. |

Escalabilidad

|  |  |
| --- | --- |
| Táctica | Descripción |
| Arquitectura en microservicios | División en servicios desacoplados para crecimiento independiente. |
| Escalado horizontal automático | Auto-scaling dinámico de recursos computacionales (contenedores/serverless). |
| Bases de datos NoSQL (grafos) | Utilizar bases de datos especializadas en almacenar grandes volúmenes de datos interrelacionados. |
| Balanceo de carga (ALB/NLB) | Distribución eficiente del tráfico de usuarios y aplicaciones externas. |

Idoneidad funcional

|  |  |
| --- | --- |
| Táctica | Descripción |
| Pruebas de aceptación automatizadas (BDD) | Validación continua contra criterios claros definidos por usuarios reales. |
| Integración continua (CI/CD) | Automatización de pruebas y despliegues para garantizar funcionalidades correctas. |
| Verificación dual humana (árbitro) | Dos transcripciones de documentos históricos + arbitraje humano para máxima precisión. |
| Auditoría y control de versiones | Capacidad de revisar, auditar y revertir fácilmente cambios en la información genealógica. |
| Feature toggles | Habilitación incremental y segura de nuevas funcionalidades o mejoras. |

# Los Patrones arquitectónicos que van a utilizar

|  |  |  |
| --- | --- | --- |
| ID | Patrón Arquitectónico | Justificación / Aplicación en el proyecto |
| P-01 | **Microservicios** | Permite modularidad y escalabilidad independiente por servicio (ej.: genealogía, búsquedas, ingestión de documentos, autenticación, moderación). |
| P-02 | **API Gateway** | Proporciona una interfaz unificada para integración segura y eficiente con terceros, gestionando autenticación, autorización, monitoreo y límites de uso. |
| P-03 | **Database per Service** | Permite usar la base de datos más adecuada por servicio: Base de datos de grafos (Neptune) para relaciones familiares y PostgreSQL para gestión documental o registros administrativos. |
| P-04 | **Cache distribuida (Redis)** | Mejora el rendimiento y eficiencia de consultas recurrentes mediante almacenamiento temporal de consultas frecuentes (subárboles familiares más populares). |
| P-05 | **Balanceador de Carga (Load Balancer)** | Asegura escalabilidad y alta disponibilidad distribuyendo eficientemente el tráfico entre múltiples instancias de microservicios. |
| P-06 | **Serverless (FaaS)d** | Uso de AWS Lambda para ejecución bajo demanda de tareas como validación de documentos, notificaciones o tareas de mantenimiento, con costos optimizados. |

# Estrategia (Táctica con los Patrones Seleccionados)

**1. Fiabilidad**

|  |  |  |
| --- | --- | --- |
| Tácticas Clave | Patrones Arquitectónicos Asociados | Descripción de la Estrategia |
| Backup & Restore automático, Redundancia de datos, Mecanismos de reintento, Monitoreo proactivo, Failover activo-activo | Microservicios, Serverless (FaaS), Balanceador de Carga | Se implementan microservicios independientes con bases de datos replicadas y backups automáticos para evitar pérdida. El balanceador de carga y failover activo aseguran alta disponibilidad. Serverless permite recuperación rápida y escalable de funciones críticas. |

**2. Seguridad**

|  |  |  |
| --- | --- | --- |
| Tácticas Clave | Patrones Arquitectónicos Asociados | Descripción de la Estrategia |
| Control de acceso granular (RBAC), OAuth 2.0 y OIDC, Cifrado en tránsito y reposo, WAF (Firewall aplicación web) | API Gateway, Microservicios | El API Gateway centraliza la autenticación y autorización mediante OAuth 2.0 / OIDC con control granular RBAC. El WAF protege el perímetro y el cifrado asegura la confidencialidad de los datos en todo momento. |

**3. Eficiencia de desempeño**

|  |  |  |
| --- | --- | --- |
| Tácticas Clave | Patrones Arquitectónicos Asociados | Descripción de la Estrategia |
| Caché distribuida (Redis), Búsqueda optimizada (índices), Replicación de lectura, CDN (CloudFront), Sharding por familias (subárbol) | Microservicios, Cache distribuida (Redis), Database per Service, Balanceador de Carga, CDN | Redis cachea consultas frecuentes para reducir latencias. Bases de datos especializadas usan índices y sharding para consultas eficientes. Balanceadores y CDN distribuyen la carga y contenido estático geográficamente. |

**4. Escalabilidad**

|  |  |  |
| --- | --- | --- |
| Tácticas Clave | Patrones Arquitectónicos Asociados | Descripción de la Estrategia |
| Arquitectura en microservicios, Escalado horizontal automático, Bases de datos NoSQL (grafos), Balanceo de carga (ALB/NLB) | Microservicios, Database per Service, Balanceador de Carga, Serverless (FaaS) | La arquitectura desacoplada permite escalar cada servicio de forma independiente. Se usa escalado automático para adaptarse a la demanda. Bases NoSQL especializadas en grafos soportan volúmenes masivos. |

**5. Idoneidad funcional**

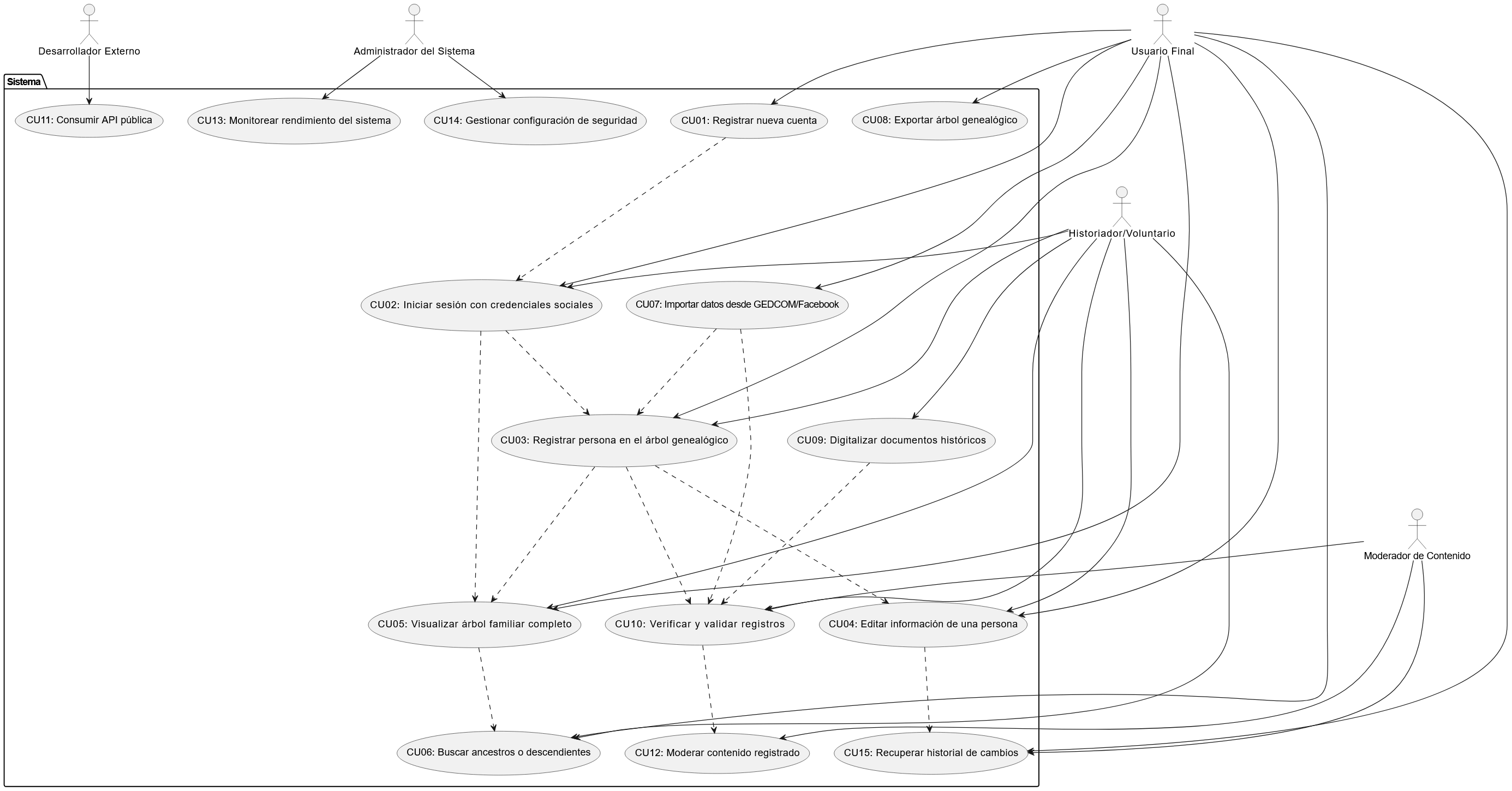
|  |  |  |
| --- | --- | --- |
| Tácticas Clave | Patrones Arquitectónicos Asociados | Descripción de la Estrategia |
| Pruebas de aceptación automatizadas (BDD), Integración continua (CI/CD), Verificación dual humana, Auditoría y control de versiones, Feature toggles | Microservicios, Serverless (FaaS), API Gateway | Se implementan pipelines CI/CD con pruebas automatizadas que validan funcionalidades críticas. Verificación dual humana y auditoría garantizan calidad y trazabilidad. Los feature toggles permiten desplegar nuevas funcionalidades gradualmente y controlar riesgos. |

La **estrategia arquitectónica** propuesta para “¿Quién es tu papá?” consiste en un sistema modular basado en **microservicios** que combinan:

* **Alta fiabilidad** y disponibilidad gracias a mecanismos automáticos de backup, redundancia y failover activos.
* **Seguridad robusta** centralizada en el API Gateway con controles detallados y cifrado.
* **Alto desempeño y baja latencia** usando caché distribuida, índices especializados y CDN para contenido estático.
* **Escalabilidad horizontal** mediante escalado automático y uso de bases de datos NoSQL orientadas a grafos.
* **Alta calidad funcional** soportada por pruebas continuas, auditorías, controles de versiones y despliegue gradual.

# Modelo 4 + 1

## 11.1 Escenarios

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

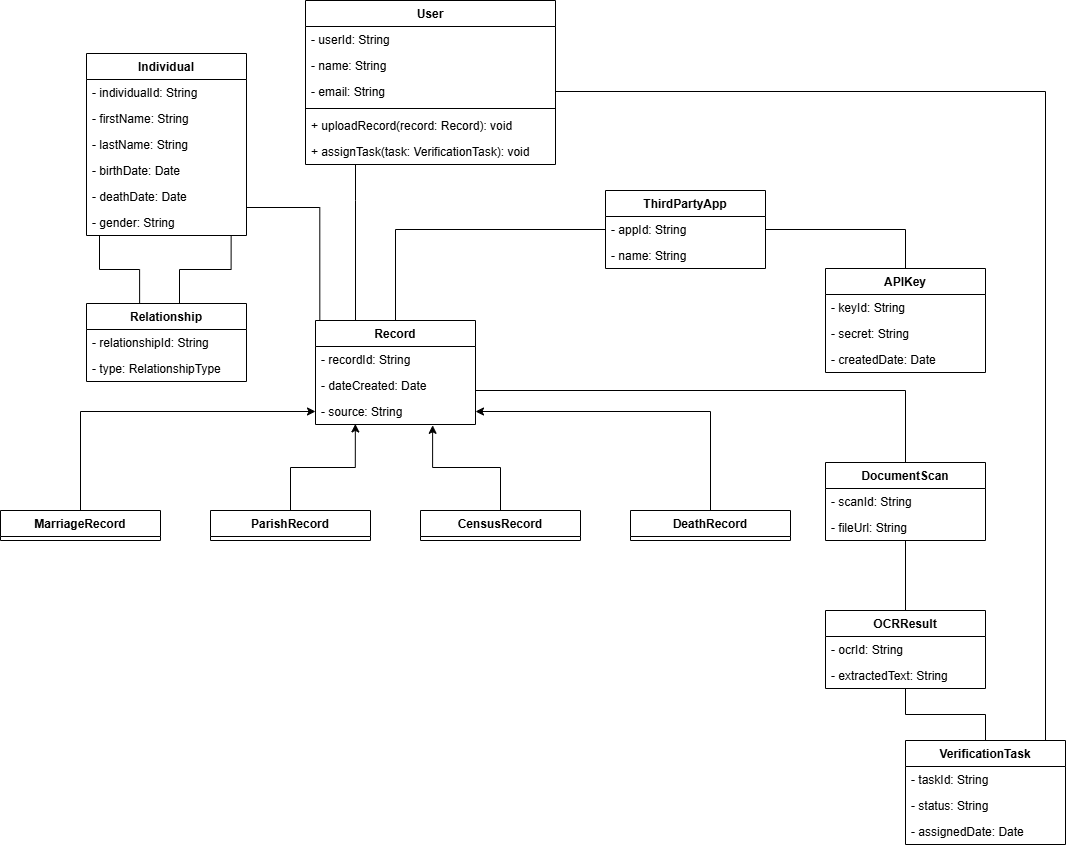
Actores

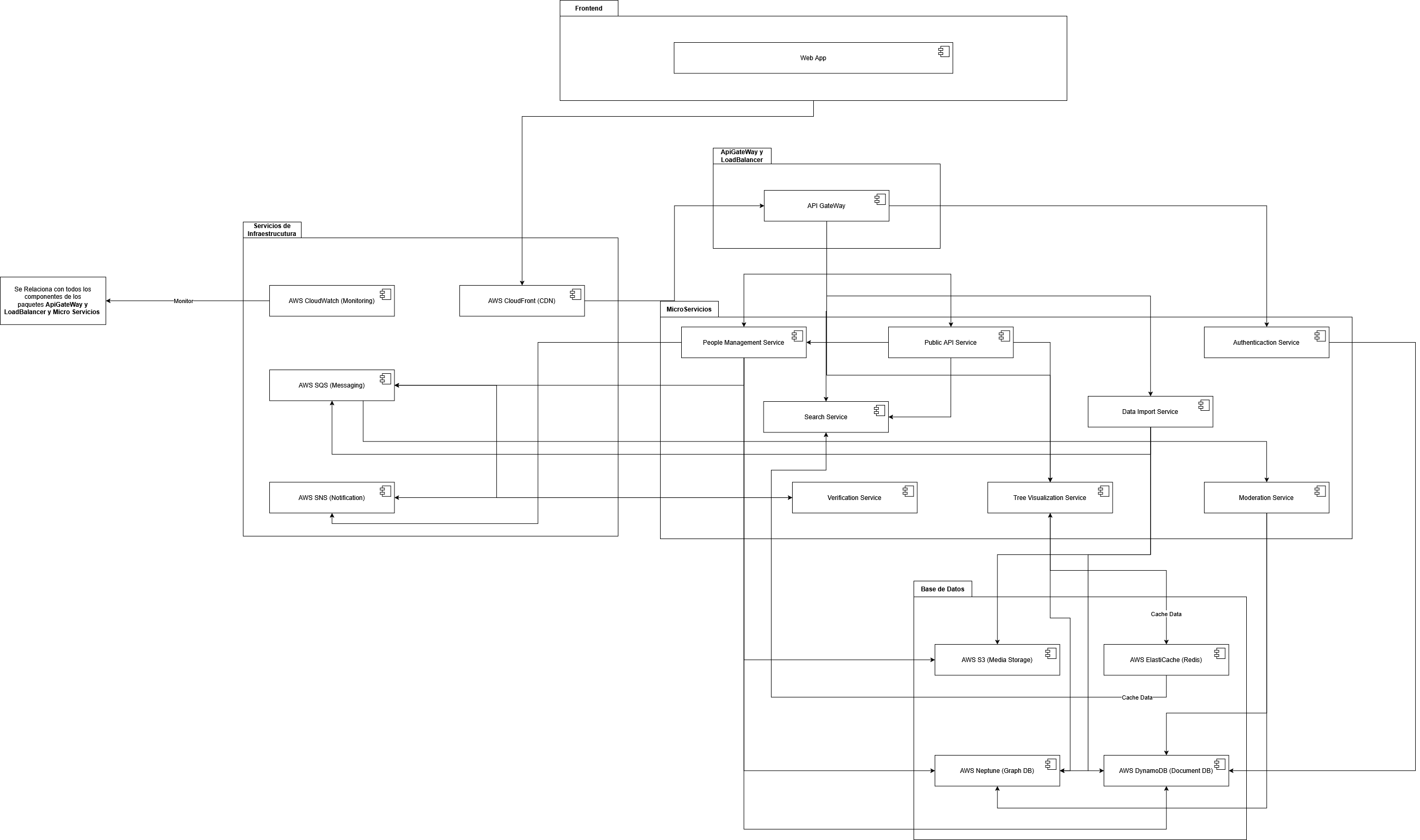
* **Usuario Final**: Personas que buscan construir su árbol genealógico.
* **Historiador/Voluntario**: Expertos que digitalizan y validan documentos históricos.
* **Desarrollador Externo**: Programadores que consumen la API pública.
* **Administrador del Sistema**: Personal técnico que gestiona la plataforma.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

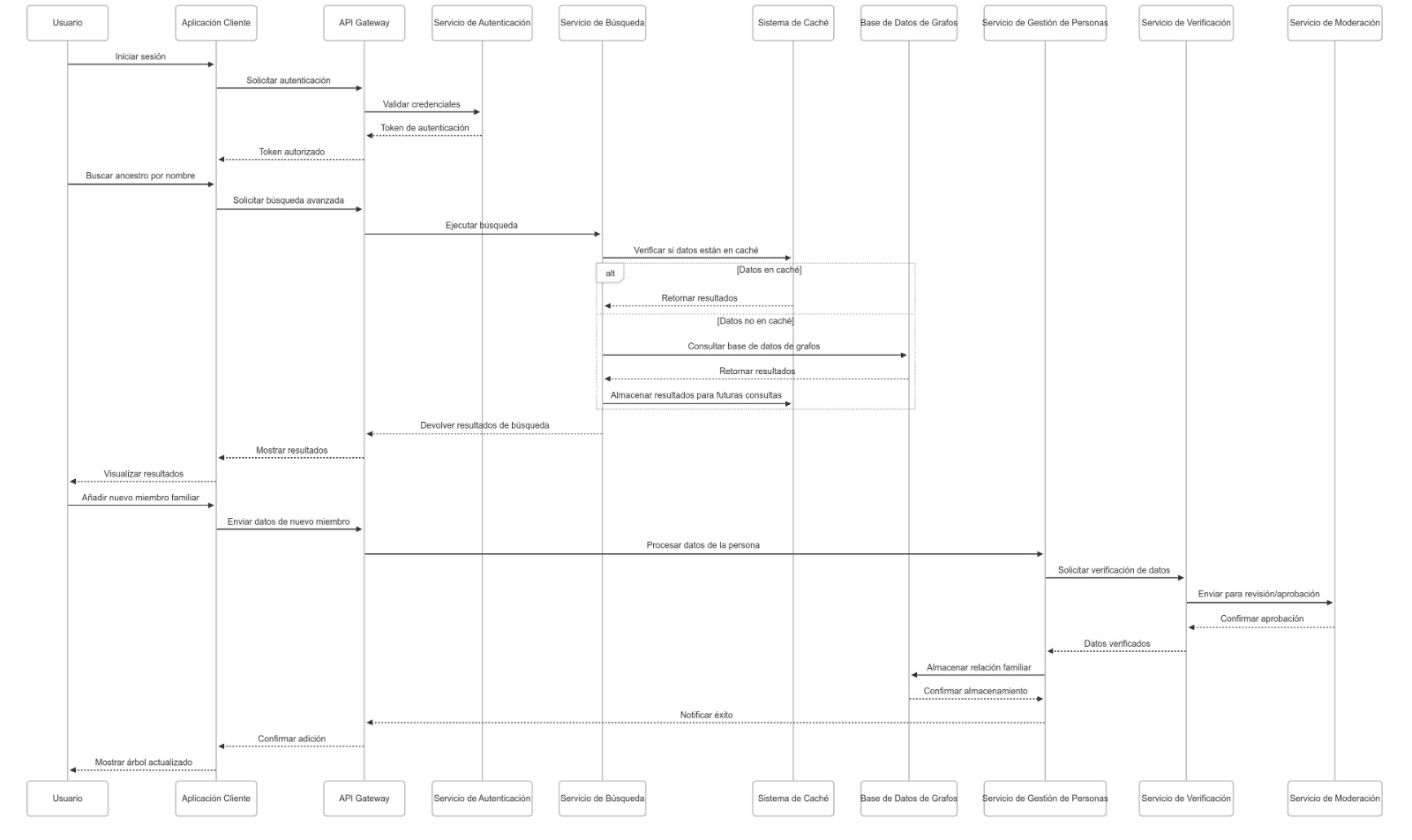
### Almacenamiento de Datos

* **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.

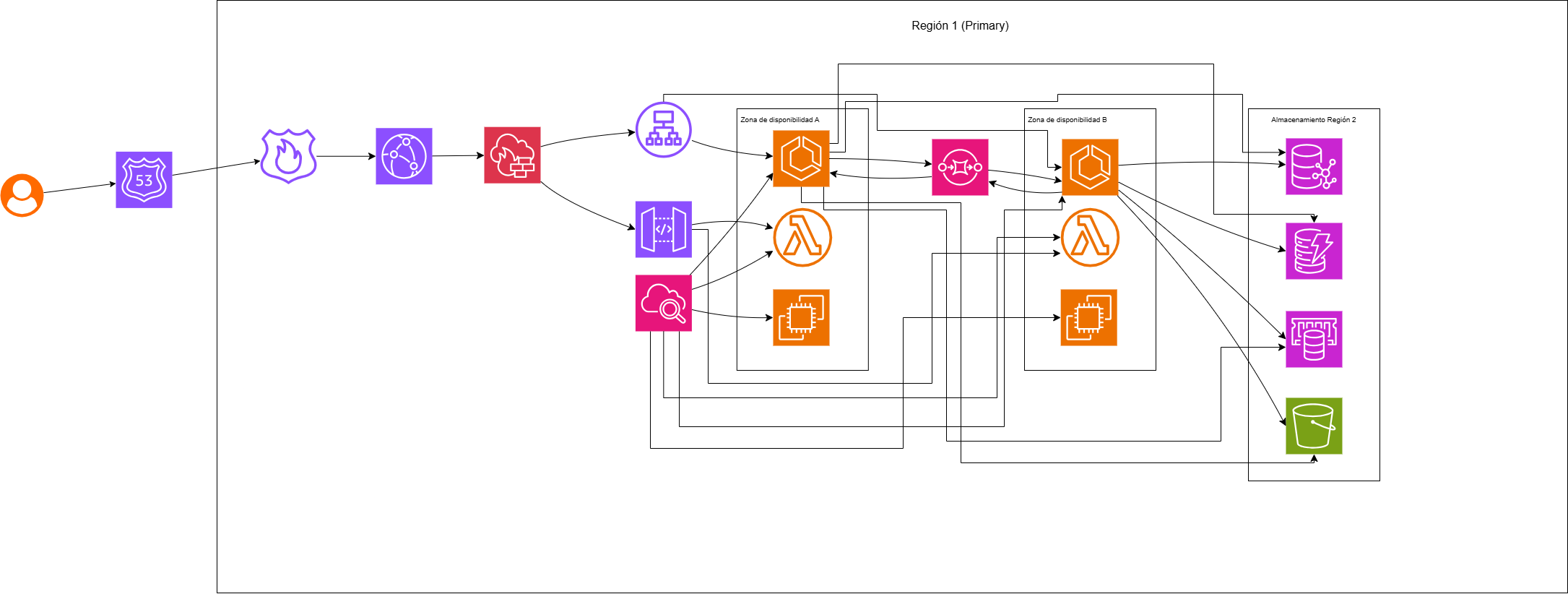
### Proceso de Búsqueda

* **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



### Multi-Región

* **Región 1 (Primaria) y Región 2 (DR)**: Implementación en múltiples regiones geográficas para alta disponibilidad.

### Por cada 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.
* **EC2 Instances**: Servidores virtuales para componentes de control.

### 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 con failover automático entre regiones.

### Almacenamiento

* **Neptune**: Base de datos de grafos con replicación entre regiones.
* **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.
* **SNS**: Sistema de notificaciones para eventos críticos.
* **CloudWatch**: Monitoreo centralizado para todos los recursos.

## 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.