Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa de Ingeniería de Sistemas

UNIVERSIDAD DEL CAUCA
DEPARTAMENTO DE SISTEMAS
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA DE SISTEMAS
DOCENTE: W. LIBARDO PANTOJA Y. 2025.1, Marzo 21 de 2025

PRIMER PARCIAL

NOMBRES Y APELLIDOS: _____ NOTA:

_____ Resolver la siguiente Kata de Arquitectura (utilizar el modelo C4): **Who's Your Daddy? ¿Quién es tu papá?**

Descripción General:

Una empresa desea construir el árbol genealógico más grande de la historia, permitiendo que millones de usuarios y aplicaciones de terceros contribuyan, visualicen y analicen datos genealógicos.

Requisitos Funcionales y no funcionales

1. El sistema debe soportar solicitudes de millones de personas que consultan, editan y agregan información genealógica. Para la determinación de parentescos, el sistema debe permitir a) El ingreso manual de relaciones por parte de los usuarios, b) Sugerencias de conexiones automáticas basadas en coincidencias con árboles genealógicos preexistentes, c) Consultar de registros históricos como censos, certificados de nacimiento/matrimonio/defunción y archivos parroquiales, d) Integración con entidades oficiales (por ejemplo, la Registraduría Nacional del Estado Civil, DANE, Archivo General de la Nación, etc), e) Algoritmos de IA y heurísticas para analizar documentos, sugerir parentescos y detectar inconsistencias.
2. Los investigadores y genealogistas podrán verificar, transcribir y enriquecer la información histórica. 3. El sistema debe tener capacidad de búsqueda avanzada, permitiendo consultar ancestros, descendientes y relaciones colaterales.
4. Resolución de conflictos de datos, identificando inconsistencias entre distintas fuentes.
5. El sistema debe permitir la generación de reportes y visualización gráfica del árbol en forma de grafo con relaciones padre-hijo, cónyuges, hermanos, primos, abuelos, etc.
6. El sistema debe ser visible y buscable en distintas plataformas (web, móvil, etc).
7. El sistema debe soportar cientos/miles de Apps que consumen y procesan datos del sistema a través de APIs. 8. El sistema debe tener capacidad para manejar millones de nodos en el grafo sin afectar el rendimiento. 9. El sistema debe tener optimización de consultas sobre grandes volúmenes de datos genealógicos.

Entregar los siguientes puntos:

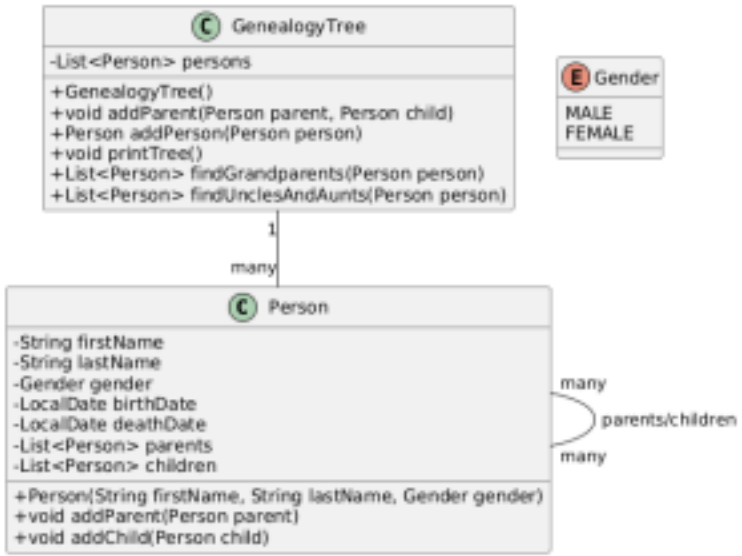
1. Diagrama de contexto: 20%
2. Diagrama de contenedores: 20%
3. Diagrama de componentes que añaden valor: 20%

4. Diagrama de clases de dos componentes importantes: 20%
5. Documentación de las principales decisiones de arquitectura usando la plantilla contexto, decisión, alternativas consideradas, consecuencias: 20%

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa de Ingeniería de Sistemas

a) Estilo(s) de arquitectura de la solución
b) Tipos de aplicaciones
c) Tecnologías principales

AYUDA. A continuación se da el diagrama de clases para el negocio de este sistema, enfocándome en los elementos claves del árbol genealógico:




```
public static void main(String[] args) {
 GenealogyTree tree = new GenealogyTree();

 Person libardo = new Person("Wilson Libardo", "Pantoja Yepez", Gender.MALE);  Person angel = new Person("Angel Libardo", "Pantoja Patiño", Gender.MALE);  Person socorro = new Person("Maria del Socorro", "Yepez Bravo", Gender.FEMALE);  Person angelica = new Person("Angelica Maria", "Bravo Arteaga", Gender.FEMALE);  Person ismael = new Person("Ismael", "Yepez Arevalo", Gender.MALE);
 Person yamile = new Person("Luz Yamilte", "Pantoja Yepez", Gender.MALE);  Person isabel = new Person("Maria Isabel", "Zambrano Pantoja", Gender.MALE);  Person juanManuel = new Person("Juan Manuel", "Pantoja Valencia", Gender.MALE);

 tree.addParent(angel, libardo);
 tree.addParent(socorro, libardo);
 tree.addParent(ismael, socorro);
 tree.addParent(angelica, socorro);
 tree.addParent(libardo, juanManuel);
 tree.addParent(yamile, isabel);
 tree.addParent(socorro, yamile);
 tree.addParent(angel, yamile);
```

```
tree.printTree();
        // Consultas específicas
System.out.println("\nAbuelos de Libardo: " + tree.findGrandparents(libardo));
```


Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa de Ingeniería de Sistemas

```
System.out.println("Tíos de Juan Manuel: " + tree.findUnclesAndAunts(juanManuel));  System.out.println("Primos de Juan
Manuel: " + tree.findCousins(juanManuel));}
```

Rúbrica de evaluación

1. Diagrama de Contexto (20%) NOTA: _____

- **Excelente (15-20 puntos):** El diagrama de contexto incluye correctamente todos los actores y sistemas externos con relaciones claras.
- **Bueno (10-14 puntos):** El diagrama de contexto incluye la mayoría de elementos pero algunos podrían mejorar en claridad y detalles importantes.
- **Aceptable (5-9 puntos):** El diagrama de contexto cumple con algunos elementos, pero tiene errores y deficiencias importantes.
- **Insuficiente (0-4 puntos):** El diagrama de contexto no cumple con los elementos requeridos o tiene errores graves.

2. Diagrama de Contenedores (20%) NOTA: _____

- **Excelente (15-20 puntos):** El diagrama de contenedores incluye todos los contenedores clave, con interacciones claras y detalladas y se especifican las tecnologías utilizadas.
- **Bueno (10-14 puntos):** El diagrama de contenedores incluye la mayoría de elementos pero algunos podrían mejorar en claridad y detalles importantes.
- **Aceptable (5-9 puntos):** El diagrama de contenedores cumple con algunos elementos, pero tiene errores y deficiencias importantes.
- **Insuficiente (0-4 puntos):** El diagrama de contenedores no cumple con los elementos requeridos o tiene errores graves.

3. Diagrama de componentes (20%) NOTA: _____

- **Excelente (15-20 puntos):** El diagrama de componentes es completo, detalla adecuadamente las interacciones entre los componentes clave y aplica correctamente los principios de diseño SOLID.
- **Bueno (10-14 puntos):** El diagrama de componentes incluye la mayoría de elementos pero algunos podrían mejorar en claridad y detalles importantes.
- **Aceptable (5-9 puntos):** El diagrama de componentes cumple con algunos elementos, pero tiene errores y deficiencias importantes.
- **Insuficiente (0-4 puntos):** El diagrama de componentes no cumple con los elementos requeridos o tiene errores graves.

4. Diagrama de clases (20%) NOTA: _____

- **Excelente (15-20 puntos):** El diagrama de clase está completo, bien estructurado y correctamente alineado con el modelo C4. Las relaciones, atributos y métodos están claramente definidos y aplica correctamente los principios de diseño SOLID.
- **Bueno (10-14 puntos):** El diagrama de clases incluye la mayoría de elementos pero algunos podrían mejorar en claridad y detalles importantes.
- **Aceptable (5-9 puntos):** El diagrama de clases cumple con algunos elementos, pero tiene errores y deficiencias importantes.
- **Insuficiente (0-4 puntos):** El diagrama de clases no cumple con los elementos requeridos o tiene

errores graves. **5. Justificación de decisiones (20%) NOTA: _____**

- **Excelente (15-20 puntos):** Se justifica claramente cada decisión arquitectónica basada en los requerimientos y restricciones del sistema y se usa la plantilla para documentar decisiones.
- **Bueno (10-14 puntos):** Algunas decisiones arquitectónicas están justificadas, pero faltan razones clave o no son claras.

- **Aceptable (5-9 puntos):** Las decisiones arquitectónicas no están justificadas adecuadamente o no se relacionan con los requerimientos.
- **Insuficiente (0-4 puntos):** No se justificaron las decisiones arquitectónicas. **Puntuación total**: _____ / 100 puntos

POSIBLE RESPUESTA KATA "QUIEN ES TU PAPA"

Diagrama de Contexto

El sistema es una plataforma web y móvil que permite la creación y consulta de árboles genealógicos a partir de información ingresada por los usuarios, registros históricos y fuentes externas como redes sociales o entidades gubernamentales.

Actores principales:

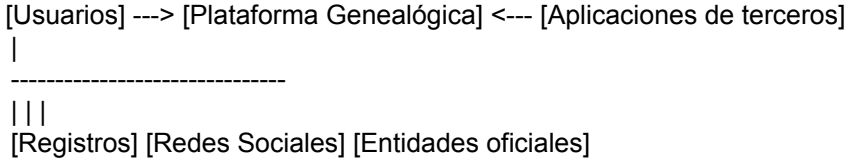
Usuarios finales: Agregan y consultan información genealógica.

Genealogistas e historiadores: Verifican y enriquecen los registros históricos.

Aplicaciones de terceros: Consumen la API pública para integrar datos genealógicos en

otros sistemas. Fuentes externas: Datos provenientes de censos, redes sociales y entidades

oficiales.



2. Diagrama de contenedores

Cada microservicio se expone a través de un API Gateway, que gestiona autenticación, autorización y balanceo de carga.

- Microservicio de Gestión Genealógica
* Maneja la creación, modificación y consulta del árbol genealógico.
* Usa una base de datos orientada a grafos (Neo4j, ArangoDB).
* Se comunica con el servicio de registros históricos para validar datos.

- Microservicio de Registros Históricos
*Almacena censos, actas de matrimonio, defunción, etc.
* Usa una base de datos documental (MongoDB, Elasticsearch).
* Procesa OCR y verificación de datos.

- Microservicio de Integración con Fuentes Externas
* Conecta con redes sociales y entidades oficiales (Facebook, Registraduría, etc.).
* Convierte y normaliza los datos antes de enviarlos al sistema.

- Microservicio de Procesamiento y Validación
* Usa algoritmos de coincidencia para sugerir relaciones familiares.
* Detecta inconsistencias y valida registros históricos.
* Puede usar IA para predecir parentescos.

- Microservicio de API para Terceros
* Exposición de datos genealógicos a otras aplicaciones.
* Implementa seguridad y control de acceso.

- Microservicio de Notificaciones y Auditoría
* Envía alertas cuando se añaden registros o se detectan errores.
* Registra cambios en la genealogía.

[Frontend Web & Móvil] --> [API Gateway] --> [Microservicio de Gestión Genealógica] --> [BD de Grafos] |
--> [Microservicio de Registros Históricos] --> [BD Documental]
|
--> [Microservicio de Integración] --> [Redes Sociales | Entidades oficiales] |
--> [Microservicio de Procesamiento] --> [IA / Validación de datos]
|
--> [Microservicio de API para Terceros]
|
--> [Microservicio de Notificaciones]

3. Componentes

Cada servicio contiene varios componentes internos. Aquí hay un desglose del Micro servicio de Gestión

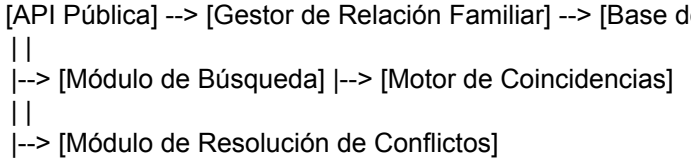
Genealógica: - Gestor de Relación Familiar: Maneja creación, modificación y eliminación de nodos en el

grafo. - Módulo de Búsqueda Avanzada:Implementa algoritmos para encontrar ancestros y descendientes.

- Motor de Coincidencias: Sugiere conexiones familiares basadas en nombres, fechas y lugares. - Módulo

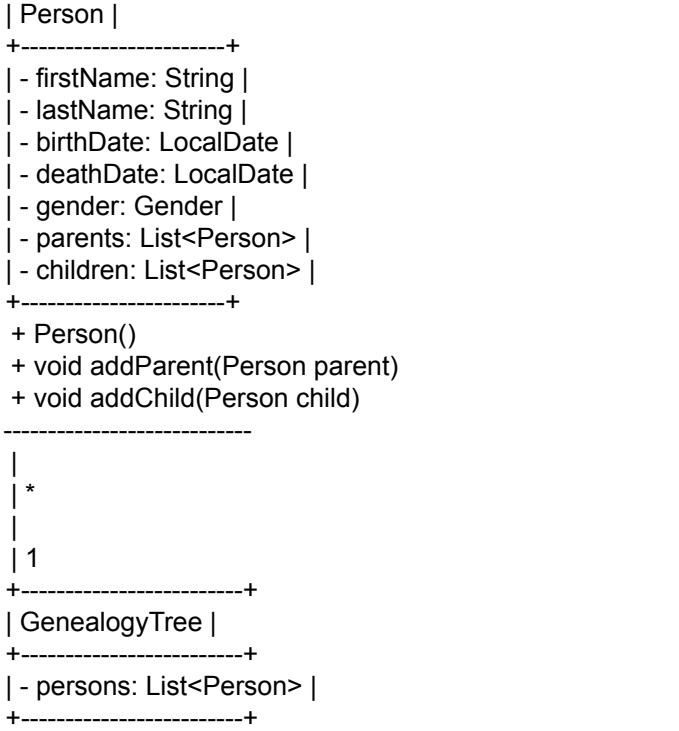
de Resolución de Conflictos: Detecta y maneja inconsistencias en los datos ingresados. - API Pública:

Expone endpoints para que aplicaciones de terceros consulten y agreguen datos.

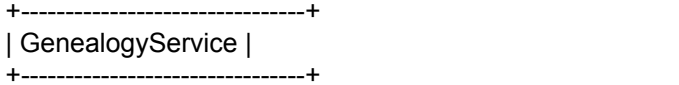


4. Diagrama clases

A continuación el diagrama de Clases del componente Gestión de Relación Familiar, Entidades:



A continuación el diagrama de Clases del componente Gestión de Relación Familiar, Services:



```
| + addPerson(person: Person) |
| + addRelation(pA, pB, relType) |
| + getAncestors(person): List<Person> |
| + getDescendants(person): List<Person> |
| + searchPerson(name): List<Person> |
+---------------------------------+

+-----------------------------------+
| RecordValidationService |
+-----------------------------------+
| + verifyRecord(record): Boolean |
| + crossCheckWithGovernmentData(record): Boolean |
| + suggestCorrections(record): List<String> |
+-----------------------------------+
```

4. DECISIONES DE ARQUITECTURA

En este ejercicios hay varias decisiones de arquitectura claves que se deben documentar:

1. El estilo arquitectónico de microservicios
2. Tipos de aplicaciones: Web / Móvil
3. Tecnologias a usar: Bases de datos, Tecnologias front, back.

A continuación mostramos la documentación del estilo de arquitectura elegido.

1. Contexto

El sistema "¿Quién es tu Papá?" debe construir y gestionar el árbol genealógico más grande de la historia. Para lograr esto, debe soportar:

Millones de usuarios y registros.

Integración con múltiples fuentes de datos (registros históricos, plataformas gubernamentales,

redes sociales). Accesibilidad desde diversas plataformas (web, móvil, API).

Escalabilidad y disponibilidad para consultas en tiempo real.

Dado que el sistema manejará una gran cantidad de datos con múltiples interacciones, se requiere una arquitectura flexible, escalable y tolerante a fallos.

2. Decisión

Se decidió adoptar una arquitectura basada en microservicios, donde el sistema se dividirá en servicios independientes que se comunicarán a través de APIs REST o mensajería. Cada microservicio se enfocará en una funcionalidad específica del dominio genealógico.

3. Alternativas Consideradas

Arquitectura Monolítica

- :) Simplicidad en el desarrollo inicial.
- :(Difícil de escalar y mantener con el crecimiento del sistema.
- :(Requiere desplegar toda la aplicación para cambios en una parte del código.
- :(Posibles problemas de rendimiento con grandes volúmenes de datos.

Arquitectura basada en Microservicios (Elegida)

- :) Escalabilidad horizontal para manejar grandes volúmenes de datos y usuarios.
- :) Mantenimiento modular, facilitando la evolución y cambios independientes.
- :) Integración flexible con múltiples fuentes de datos.
- :(Mayor complejidad en la gestión de despliegues y comunicación entre servicios.

Arquitectura Basada en Servicios (SOA clásica)

- :) Permite modularidad y reusabilidad.
- :(Mayor dependencia de un Enterprise Service Bus (ESB), lo que introduce un punto único de fallo. • :(No proporciona la escalabilidad granular que se necesita para este proyecto.

4. Consecuencias
Positivas:

Se mejora la escalabilidad y disponibilidad del sistema.

Se facilita la integración con múltiples fuentes de datos y plataformas.

Se permite la evolución del sistema sin afectar toda la aplicación.

Se mejora la resiliencia al fallo de un servicio sin afectar otros módulos.

Negativas:

Mayor complejidad en la gestión de despliegues y monitoreo.

Se requiere un mecanismo de comunicación eficiente entre microservicios (mensajería o API

Gateway). Posible aumento en la latencia debido a llamadas de red entre servicios.