



EduGrade global

SISTEMA NACIONAL DE CLASIFICACIONES MULTIMODELO

Daniela Gangotena
Alejandro Valente

Estrategia





Tolerancia a fallos

Con N=3:

El sistema puede perder hasta 2 nodos

Mientras quede al menos 1 nodo activo:

Se cumple W=1

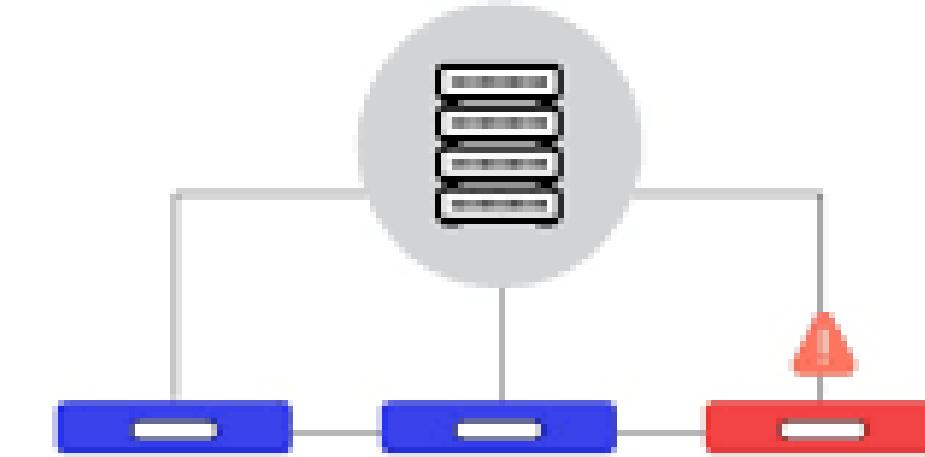
Se cumple R=1

El sistema sigue operando sin interrupción

Esto cumple el requisito:

Puede perder 2 nodos y continuar funcionando.

Tolerancia a fallos



Arquitectura:

Cantidad de Clusters

3 cluster principal de 1 nodo

Clusters Regionales

Particionamiento (Sharding)

Sharding por 3 regiones (sudafrica dividido en 3)

Shard 1 → capital (Johannesbourg)

Shard 2 → sur (Cape Town)

Shard 3 → este (Durban)

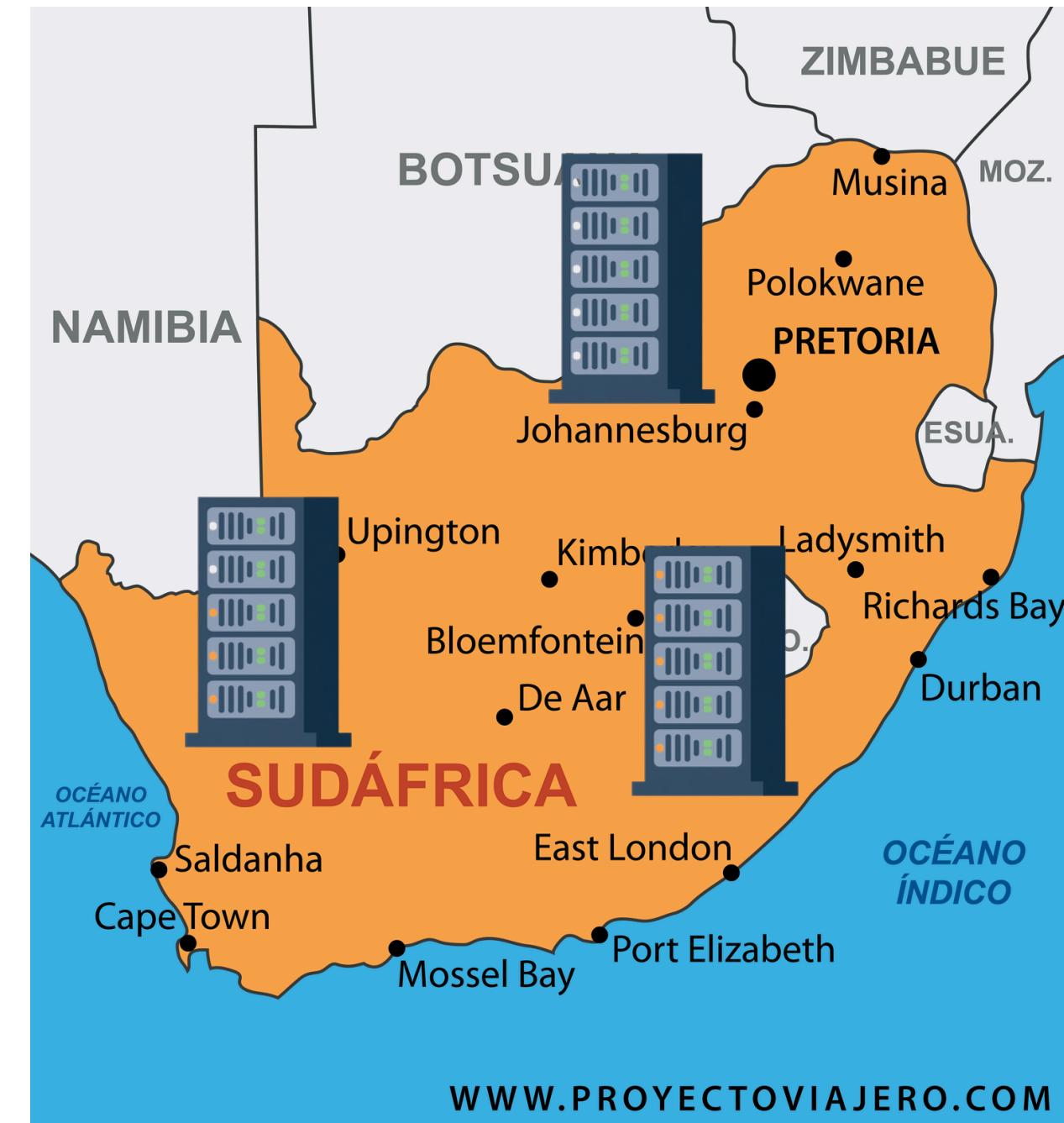
Ventajas:

- Reduce latencia regional
- Mejora escalabilidad horizontal
- Permite balanceo de carga

Distribución de carga

- **Con 3 nodos:**

- Cada nodo almacena aproximadamente 1/3 de los shards
- Replicación cruzada entre nodos
- Balanceador distribuye lecturas y escrituras



Negocio Continuo

- **El sistema garantiza:**
 - Escuelas pueden seguir cargando notas
 - Consultas de estudiantes siempre disponibles
 - Operación 24/7
 - No se detiene ante fallos regionales
- **Ideal para:**
 - Períodos de matrícula
 - Cierre de trimestres
 - Publicación de calificaciones



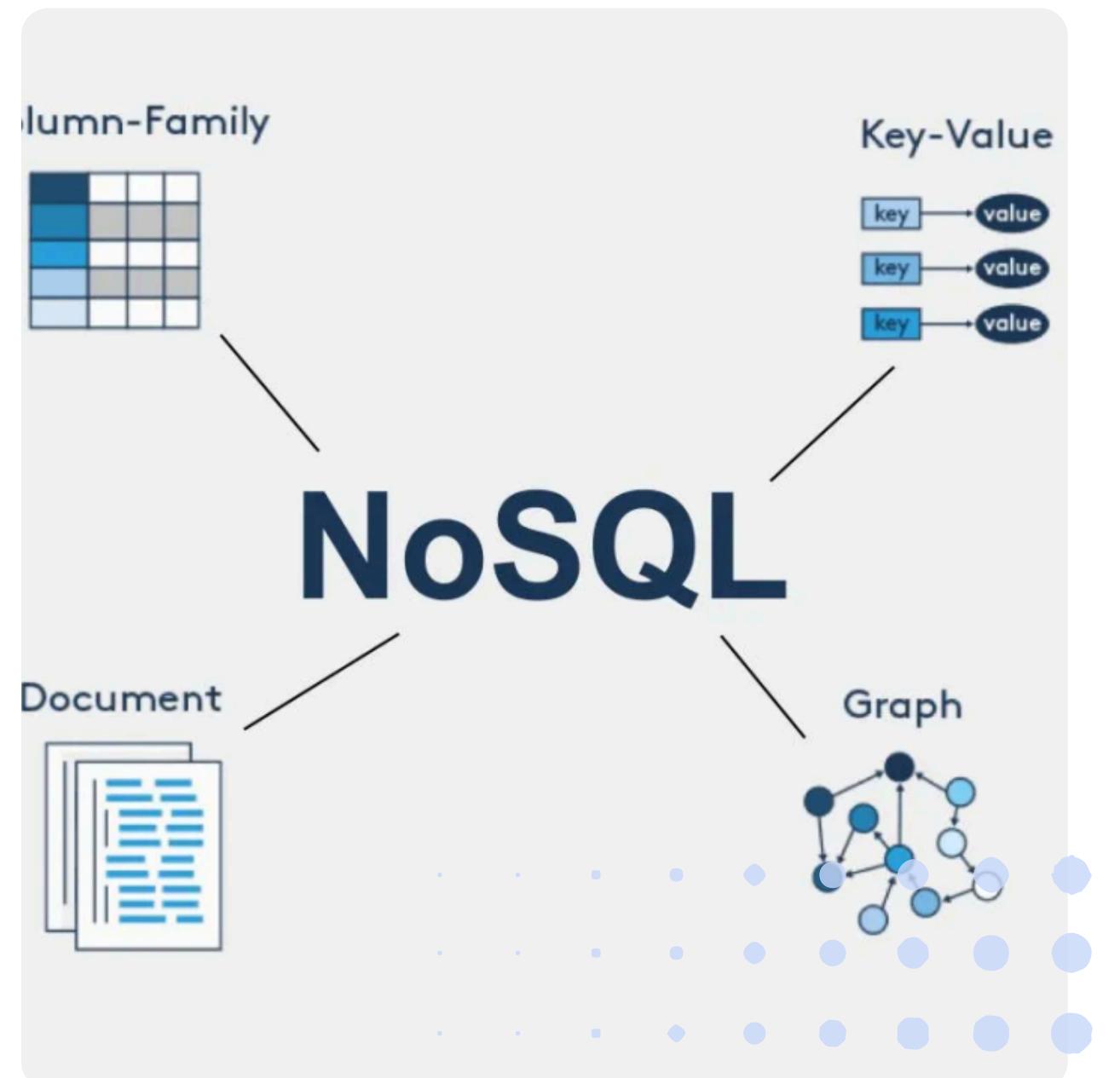
Modelo

- **Bajo:**

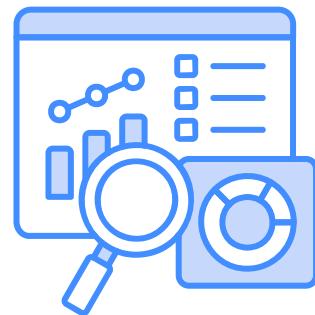
- presupuesto
- flujo de entrada y salida de datos
- picos de actividad (Períodos de matrícula,Cierre de trimestres,Publicación de calificaciones)

- **Al ser modelo AP:**

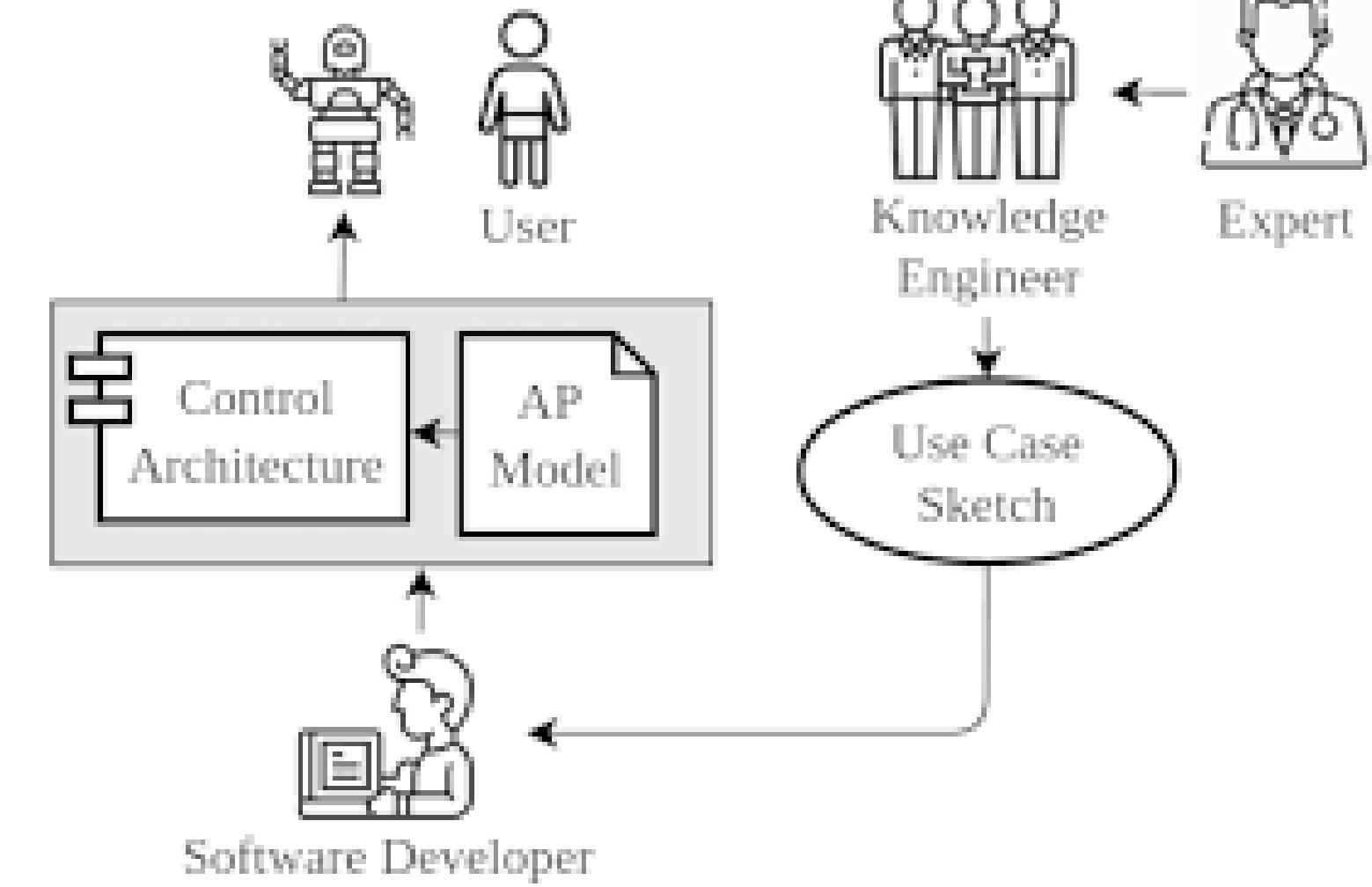
- Puede existir inconsistencia temporal
- Un nodo puede tener una nota actualizada y otro no
- Eventualmente todos los nodos convergen



Ventajas



**Nunca se bloquea el sistema
No se rechazan solicitudes
El sistema no se detiene**

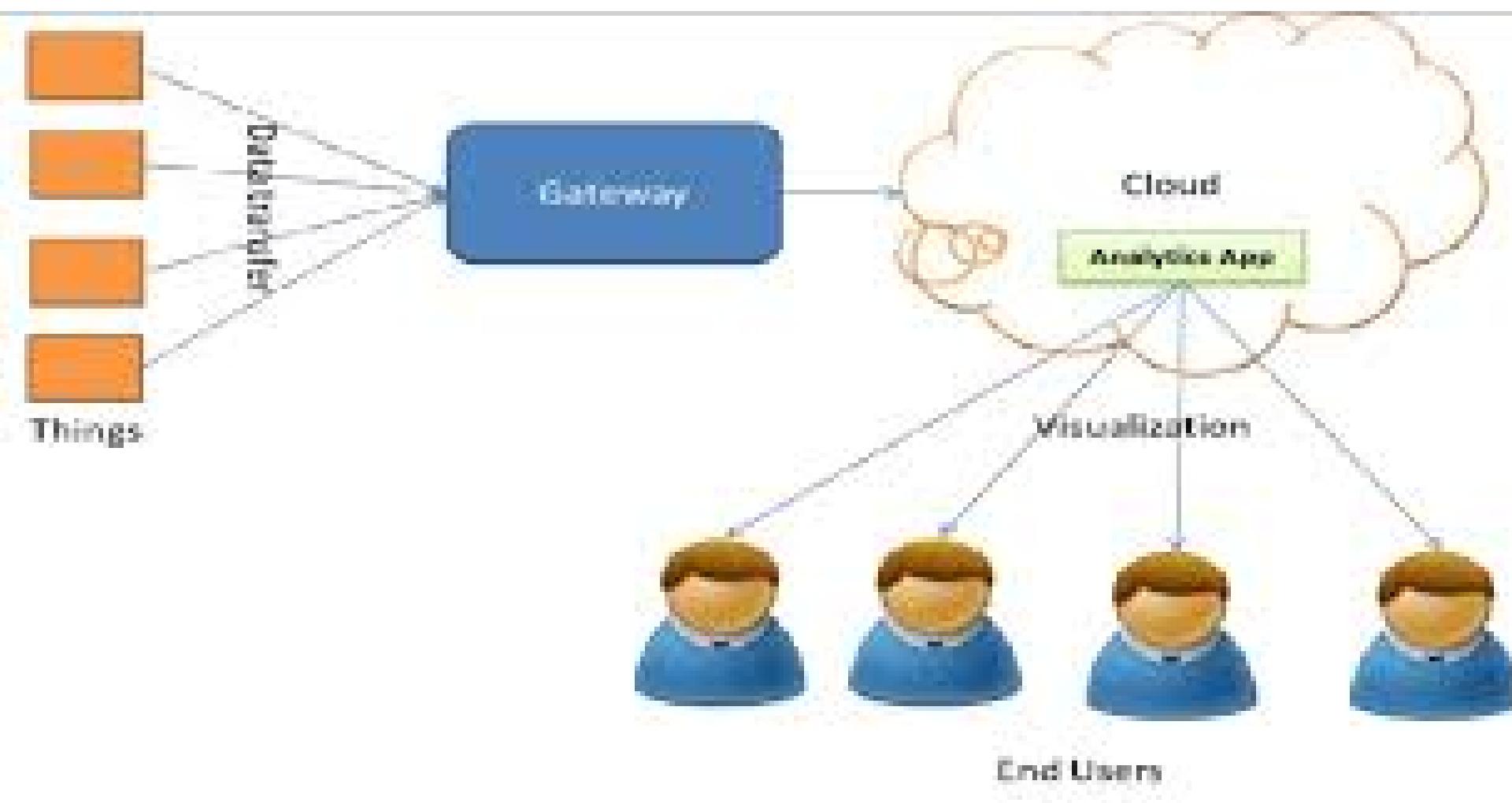


En un sistema educativo nacional, la disponibilidad es más crítica que la consistencia estricta inmediata.

Consideraciones

Si el sistema educativo crece:

- Se agregan más nodos al cluster
- Se redistribuyen shards
- Se puede pasar de 3 a 5 nodos por clusters
- Se pueden agregar nuevos clusters
- Escalabilidad horizontal sencilla



Estructura del proyecto



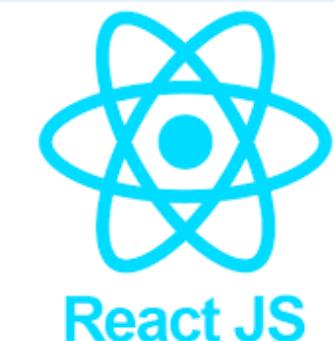
Base de datos no sql



Contenedor con imagenes

	Name	Container ID	Image	Port(s)
<input type="checkbox"/>	demo	-	-	-
<input type="checkbox"/>	redis_db	4e3f526287a3	redis:latest	6379:6379
<input type="checkbox"/>	neo4j_db	1fa218e92256	neo4j:latest	7474:7474
<input type="checkbox"/>	cassandra_db	00ea91b09004	cassandra:latest	9042:9042

Tecnologías



Demostración en Vivo

**dggtn/
ingenieria_datos2_TPO**



Sistema de bases de datos distribuidas

2

Contributors

0

Issues

0

Stars

0

Forks



dggtn/ingenieria_datos2_TPO: Sistema de bases de datos distribuidas

Sistema de bases de datos distribuidas . Contribute to

dggtn/ingenieria_datos2_TPO development by creating an account on GitHub.





Mongo db

1

MongoDB(CP): Consistencia y Particionamiento.

La elegimos para el Registro de Notas porque no podemos permitir datos inconsistentes en el origen de la transacción.



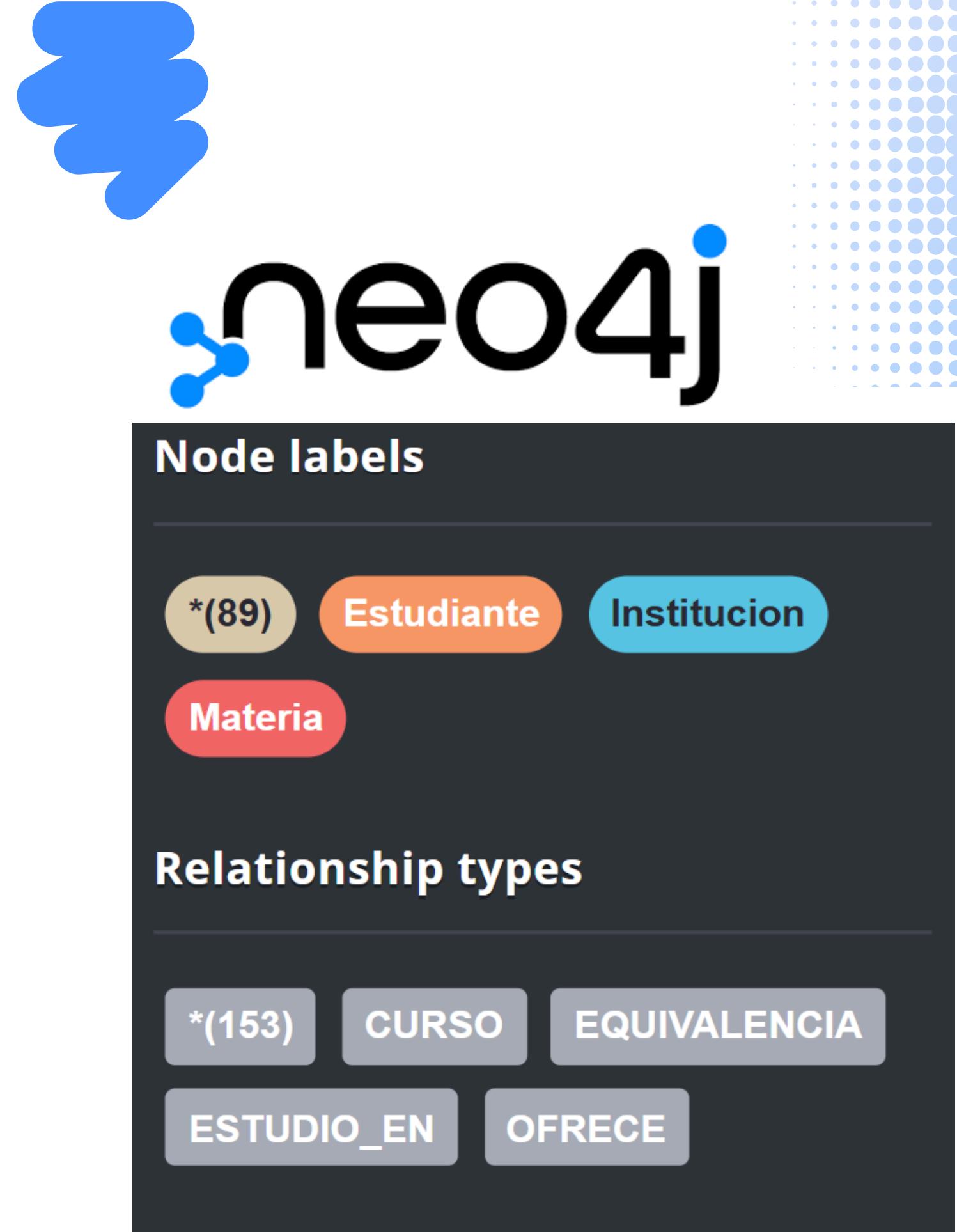
Collections

				Collection Name	Create collection
				calificaciones	
				estudiantes	
				legislaciones_conversion	



Neo4J

- 1 Neo4j (CA/CP): Alta disponibilidad y consistencia en relaciones. Ideal para las relaciones académica, donde importa más el "vínculo" (Estudiante-Materia-Instituto) que el dato aislado.





Cassandra

1 Cassandra (AP): Disponibilidad y Tolerancia al Particionamiento. Perfecta para los Reportes Masivos, donde el sistema debe responder siempre, aunque el dato tenga milisegundos de retraso (consistencia eventual).



2

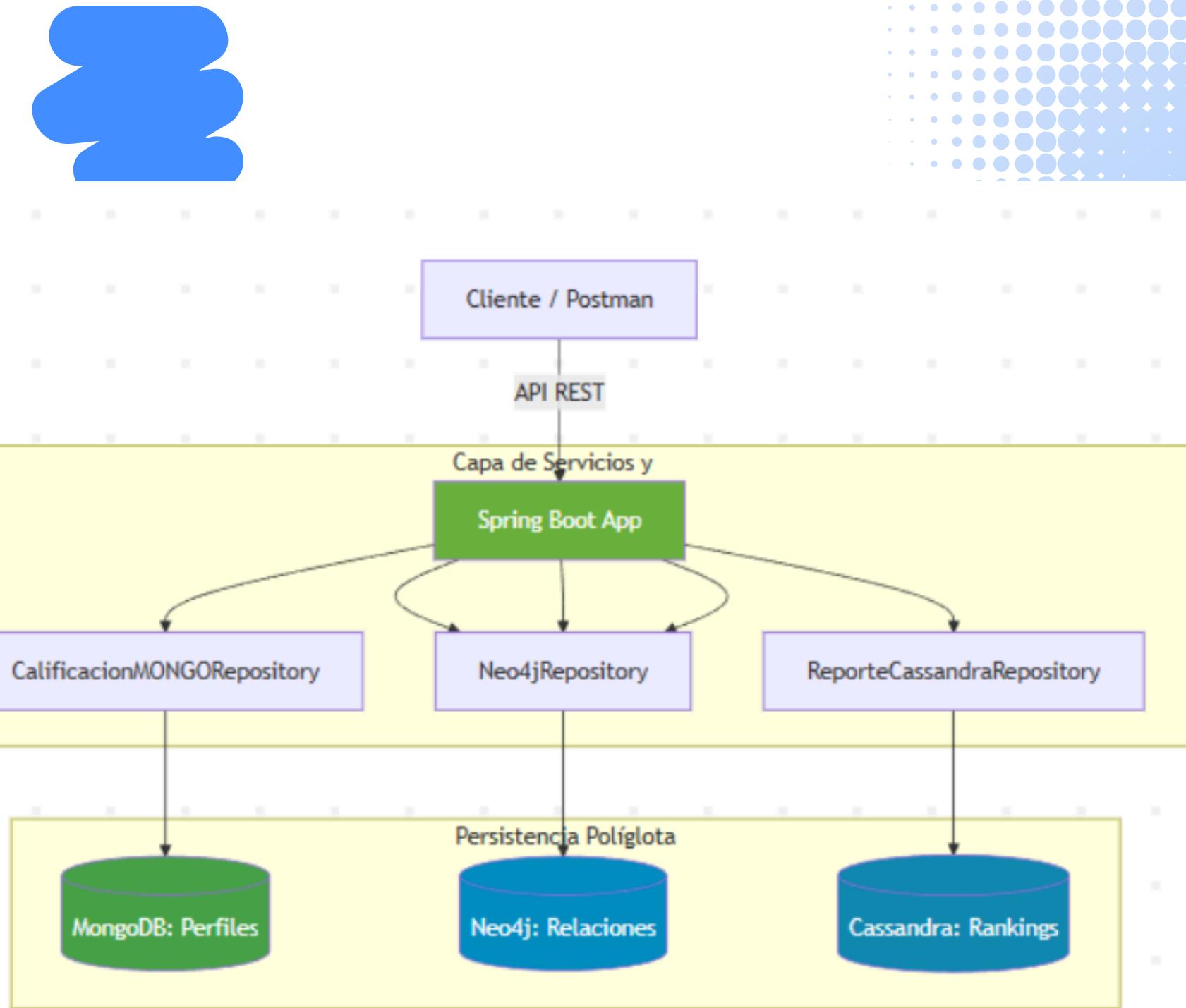
```
@Repository 6 usages ✎ Daniela
public interface ReporteCassandraRepository extends CassandraRepository<Reporte, String> {
    @Query("SELECT * FROM reporte WHERE tipo = ?0 LIMIT ?1") 2 usages ✎ Daniela
    ⚡ List<Reporte> obtenerTopPorTipo(String tipo, int limite);
}
```

```
@Repository 6 usages ✎ alvalenteUADE *
public interface CalificacionCassandraRepository extends CassandraRepository<CalificacionCassandra, String> {
    @Query("SELECT institucionid, institucionnombre, institucionpais, institucionprovincia, institucionniveleducativo, " +
        "" + "estudianteid, notaconversionssudafrica FROM calificaciones")
    List<CalificacionCassandra> obtenerDatosParaRankingInstituciones();
```



Arquitectura y Flujos

- 1. El usuario registra en el Frontend.**
- 2. Spring Boot guarda el documento en MongoDB.**
- 3. Se crea la relación en el grafo de Neo4j.**
- 4. Un proceso de sincronización (Batch) impacta los promedios en Cassandra.**



Se usa un modelo Schema-less en Mongo para flexibilidad y un modelo Query-first en Cassandra

Desempeño y Escalabilidad



Optimización de Lectura en Cassandra:

A diferencia de los sistemas RDBMS tradicionales que dependen de costosas operaciones de agregación (AVG, SUM, GROUP BY) en tiempo de ejecución, nuestra arquitectura en Cassandra utiliza un enfoque de Modelado Orientado a Consultas (Query-First Design). Logramos una latencia de lectura de orden constante $O(1)$ por partición, permitiendo la generación de rankings sobre millones de registros sin penalizar el tiempo de respuesta del sistema.



Estrategia de Particionamiento:

Implementamos una Partition Key basada en las 3 regiones para asegurar una distribución uniforme de los datos en el clúster. Esto **evita los "Hotspots"** (nodos sobrecargados) y permite un escalado horizontal lineal: a mayor volumen de datos, simplemente se añaden nodos al anillo de Cassandra sin reestructurar la lógica de la aplicación.



Trade-offs (CAP):

Se tomó la decisión de diseño de **priorizar la Disponibilidad y Tolerancia al Particionamiento (AP)** en el módulo de reportes. Aplicamos un **modelo de Consistencia Eventual**: sacrificamos la consistencia inmediata (linearizabilidad) a cambio de una Alta Disponibilidad y una velocidad de respuesta instantánea en el Dashboard. Esto garantiza que el sistema siempre sea capaz de procesar nuevas calificaciones, delegando la sincronización de promedios a un proceso asíncrono configurando LOCAL_QUORUM



Trade-offs y Decisiones de Diseño

"Implementar una arquitectura de Persistencia Políglota introduce desafíos que fueron evaluados y mitigados durante el desarrollo del proyecto"

Desafío: Sincronización de Datos (Eventual Consistency)

Trade-off: Al tener tres motores distintos, el dato no llega a todos al mismo tiempo.

Decisión: Aceptamos una latencia de propagación.

Priorizamos la respuesta inmediata en el registro (MongoDB) y delegamos la actualización del Grafo (Neo4j) y los Reportes (Cassandra) a procesos asíncronos. Esto evita el "bloqueo" del sistema ante fallos parciales.

Trade-offs y Decisiones de Diseño

Desafío: Duplicidad de Información (Redundancia Controlada)

Trade-off: La entidad de estudiante ,intsitucion, materia y el promedio residen en las tres bases de datos.

Decisión: Optamos por la Desnormalización. El costo de almacenamiento es barato comparado con el costo computacional de realizar cruces de datos (Joins) entre nodos distribuidos. Ganamos velocidad de lectura sacrificando espacio en disco.

Trade-offs y Decisiones de Diseño

Desafío: Complejidad Operativa (Mantenibilidad)

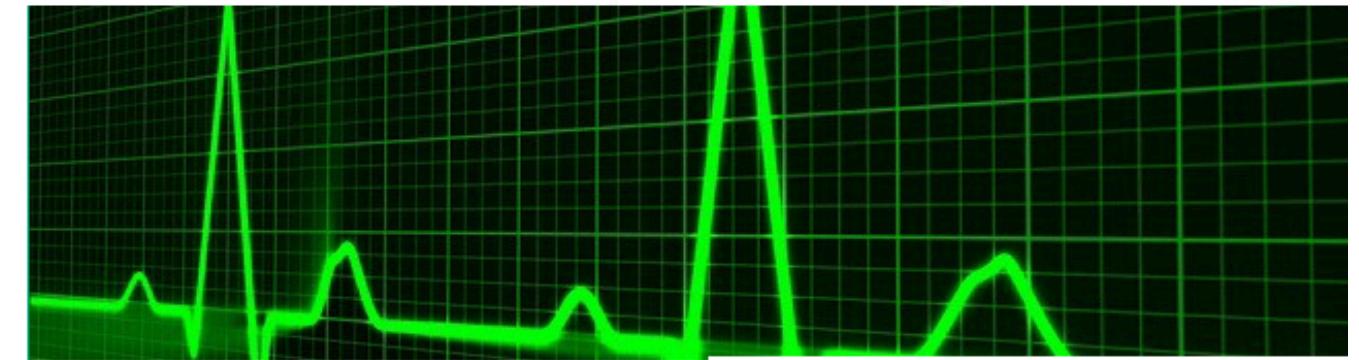
Trade-off: Mantener tres esquemas y conexiones diferentes aumenta la curva de aprendizaje y el mantenimiento.

Decisión: Utilizamos Spring Data como capa de abstracción unificada y Docker Compose para la orquestación. Esto nos permite gestionar la infraestructura como código, facilitando el despliegue y la escalabilidad del stack tecnológico.

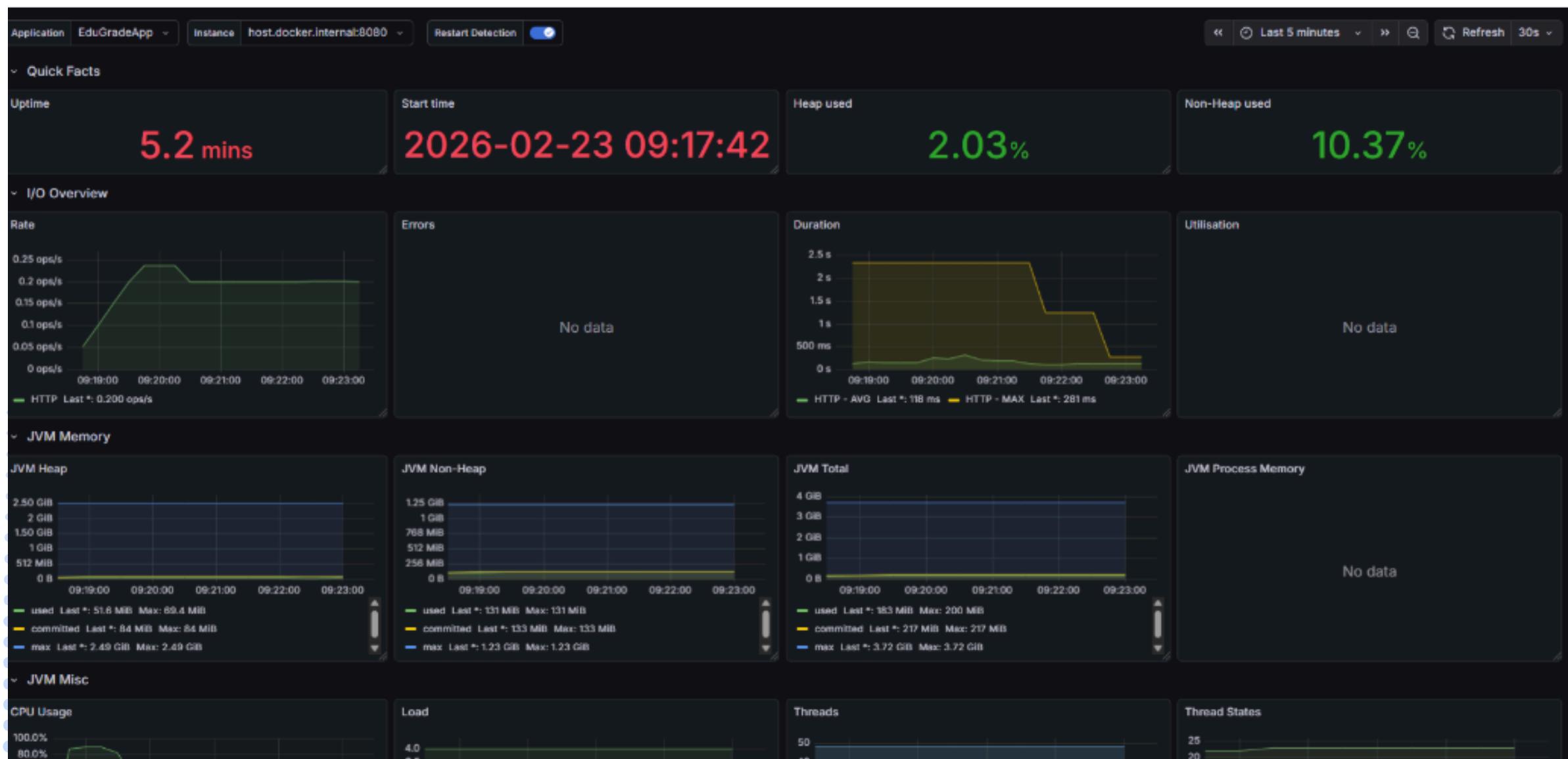
Arquitectura y Observabilidad



- Stack: Micrometer + Prometheus + Grafana.
- Propósito: Seguimiento en tiempo real de la salud del backend y el rendimiento de las consultas a Neo4j y Cassandra.



Panel de Control de Infraestructura.



- **Uso de Heap Memory:** Control de consumo de RAM al procesar grandes volúmenes de datos.
- **Latencia de Endpoints:** Tiempo de respuesta de las conversiones de notas.
- **Tasa de Errores:** Monitoreo de excepciones en las conexiones NoSQL.



**Mientras las bases de datos
guardan los datos, Grafana nos
permite visualizar el flujo de
trabajo y detectar cuellos de
botella en la sincronización de
datos**

Swagger

calificacion-controller	
PUT	/api/calificaciones/{id}
POST	/api/calificaciones/similar-conversion
POST	/api/calificaciones/registrar
reporte-controller	
POST	/api/reportes/promedios
GET	/api/reportes/top-paises
GET	/api/reportes/top-institutos
GET	/api/reportes/provincias-ranking
GET	/api/reportes/niveles-educativos-ranking
GET	/api/reportes/instituciones-ranking
materia-controller	
POST	/api/materias/registrar
POST	/api/materias/equivalencias
GET	/api/materias/{idMateria}/equivalencias
GET	/api/materias/opciones
GET	/api/materias/instituciones/{idInstitucion}/opciones
institucion-controller	
POST	/api/instituciones/registrar
GET	/api/instituciones/{id}
GET	/api/instituciones/opciones
estudiante-controller	
POST	/api/estudiantes/registrar
GET	/api/estudiantes/{id}/instituciones
GET	/api/estudiantes/{id}/detalle-completo
GET	/api/estudiantes/opciones

DockerHub

	Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Memory (%)	Disk read/w...	Actions
<input type="checkbox"/>	demo	-	-	-	N/A	N/A	N/A	N/A	 
<input type="checkbox"/>	prometheus	c5bb5eed877a	prom/prom	9090:9090	N/A	N/A	N/A	N/A	 
<input type="checkbox"/>	grafana	b8c9f01511f9	grafana/grafana	3001:3000	N/A	N/A	N/A	N/A	 
<input type="checkbox"/>	mongo-expre...	994f4103f705	mongo-express	8081:8081	N/A	N/A	N/A	N/A	 
<input type="checkbox"/>	cassandra_dt	0b9073a65c39	cassandra:cassandra	9042:9042	N/A	N/A	N/A	N/A	 
<input type="checkbox"/>	mongodb	58c3fcd9459c	mongo:latest	27017:27017	N/A	N/A	N/A	N/A	 
<input type="checkbox"/>	neo4j	286b97946213	neo4j:latest	7474:7474 Show all ports (2)	N/A	N/A	N/A	N/A	 