DataCat

Contexto

Datacat es una plataforma de monitoreo y análisis en la nube diseñada para ayudar a las organizaciones a supervisar el rendimiento de sus sistemas, aplicaciones y servicios en tiempo real. A través de la recopilación de métricas, logs y trazas de diversas fuentes, Datacat permite a los equipos de desarrollo y operaciones identificar rápidamente problemas de rendimiento, detectar errores y optimizar la infraestructura. Con una interfaz visual intuitiva y funciones avanzadas de análisis, Datacat facilita la observación continua de entornos complejos, ofreciendo insights valiosos para mejorar la eficiencia y asegurar una experiencia de usuario óptima.

En esta oportunidad, nos enfocaremos en la recopilación y consulta de métricas, no al stack completo de observabilidad.

Organizaciones

Nuestros clientes son organizaciones, que poseen una o más aplicaciones, las cuales quieren monitorear, para eso nos envían métricas a nosotros. Cada aplicación no envía la métrica directamente, sino que posee un recolector, el cual funciona como buffer y luego se envían batches a DataCat.

Métricas

Una métrica representa a un conjunto de datos cuantitativos (números) que describen un aspecto del comportamiento, estado o rendimiento de un sistema, aplicación o infraestructura. Las métricas se usan para monitorear, analizar y optimizar operaciones.

Medición

Es el valor de una métrica, en un momento determinado (timestamp) y una serie de tags asociados. Los tags (o etiquetas) son pares clave-valor(string, string) que se añaden a las métricas para clasificarlas, filtrarlas o agruparlas según dimensiones específicas, como la región, el nombre del servicio o el entorno (ej. region=us-east-1, env=prod, client=gativideo, app=web-gativideo, status=200). Esto permite un análisis más detallado y flexible. Tiene al menos los tags que identifican al cliente / organización y la aplicación del mismo.

Componente gráfico y consulta

Para observar las métricas, la forma más eficiente de hacerlo es mediante gráficos o tablas. Vamos a tener varios tipos de gráficos, pero por ahora solo están dentro del alcance las series de tiempo (tiene un ejemplo más abajo). Lo que se grafica en el componente son las mediciones de una métrica, agrupadas mediante alguna función, para un momento dado, y pueden ser filtrados a partir de tags. La selección de la métrica, la función de agrupación (promedio, suma, percentil X) y filtro se denomina *consulta*. Por otro lado, también se pueden graficar relaciones entre mediciones, a lo cual llamaremos *fórmulas*. Asimismo, una *fórmula*, puede no solo depender de los valores de las *consultas*, sino también de otras *fórmulas*. La función de agrupación y los filtros son ejecutados en la TSDB.

Cada gráfico tiene un título y una descripción y puede ser utilizado en distintos tableros. Cada consulta y cada fórmula tienen un color dentro del componente y pueden dibujarse o no.

Tablero

Un tablero tiene un nombre, fecha de modificación, y uno o más componentes gráficos. Cada componente gráfico tiene una posición y un tamaño dentro del tablero. Cuando se visualiza el tablero, se debe seleccionar el intervalo de tiempo correspondiente y todos los componentes gráficos muestran los resultados sobre dicho intervalo.

Monitor

Los mismos se crean para vigilar los valores de una consulta o fórmula, de manera periódica. La vigilancia puede consistir en ver si las consultas o fórmulas:

- Están por debajo/arriba de un valor o umbral
- Ver si las mediciones están por debajo/arriba de un valor o umbral, dentro de un intervalo de tiempo.

Si se cumple alguna de las condiciones anteriores, se debe activar una alarma, que consiste en informar de la situación a los clientes, en principio mediante Slack o Telegram. Cada cliente selecciona para cada monitor una o más formas de notificar y las credenciales se configuran a nivel organización.

Consideraciones IMPORTANTES

- Los clientes son los que crean las entidades (métricas, tableros, monitores, etc) mencionadas en el punto anterior, y no se pueden compartir entre clientes.

- Las consultas a la TSDB se realizan periódicamente en los monitores y cada vez que se visualiza un tablero (o sea, cuando el front lo solicita).

Punto 1 – Arquitectura (40 puntos)

Parte A: Actualice el diagrama de Despliegue del enunciado (y realice cualquier diagrama o explicación auxiliar que necesite) teniendo en cuenta lo siguiente:

1. **(5 puntos)** Para el Front-End, queremos que los usuarios finales (que pertenecen a alguna organización/cliente) puedan conectarse con sus propias credenciales (es decir, no deben necesitar un usuario/pass en DataCat). De un pequeño ejemplo

2. **(15 puntos)** Dado que DataCat-Jobs, es despertado por un un CRON/Tarea programada que se ejecuta cada minuto, explique cómo permitir que escale la ejecución de los monitores (debe suponer que la TSDB escala adecuadamente con las consultas). - EXPLIQUE el funcionamiento (5/15) y de un ejemplo (5/15)

3. **(15 puntos)** Cada vez que hay un problema en un cliente (por ejemplo una de sus aplicaciones empieza a fallar o a tardar mucho en contestar pedidos), lo que suele pasar es que decenas de sus usuarios, van a ver los mismos tableros durante intervalos de una hora al menos, hasta que se resuelve el problema. ¿Detecta alguna oportunidad de mejora desde la parte arquitectónica? EXPLIQUE el funcionamiento (5/15) y de un ejemplo (5/15)

Parte B: (5 puntos) para el front end, se barajó la posibilidad de que en lugar de que sea una aplicación web de cliente pesado, la misma sea un cliente liviano (render server). Teniendo en cuenta su experiencia mirando estos tableros, está de acuerdo con la implementación? compare / justifique.

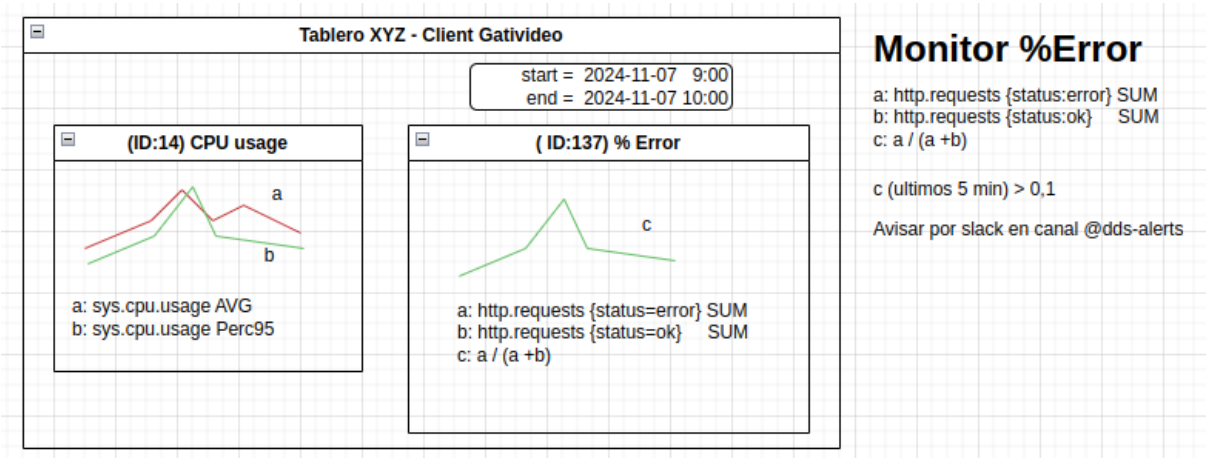
Punto 2 - Modelado de dominio (35 puntos)

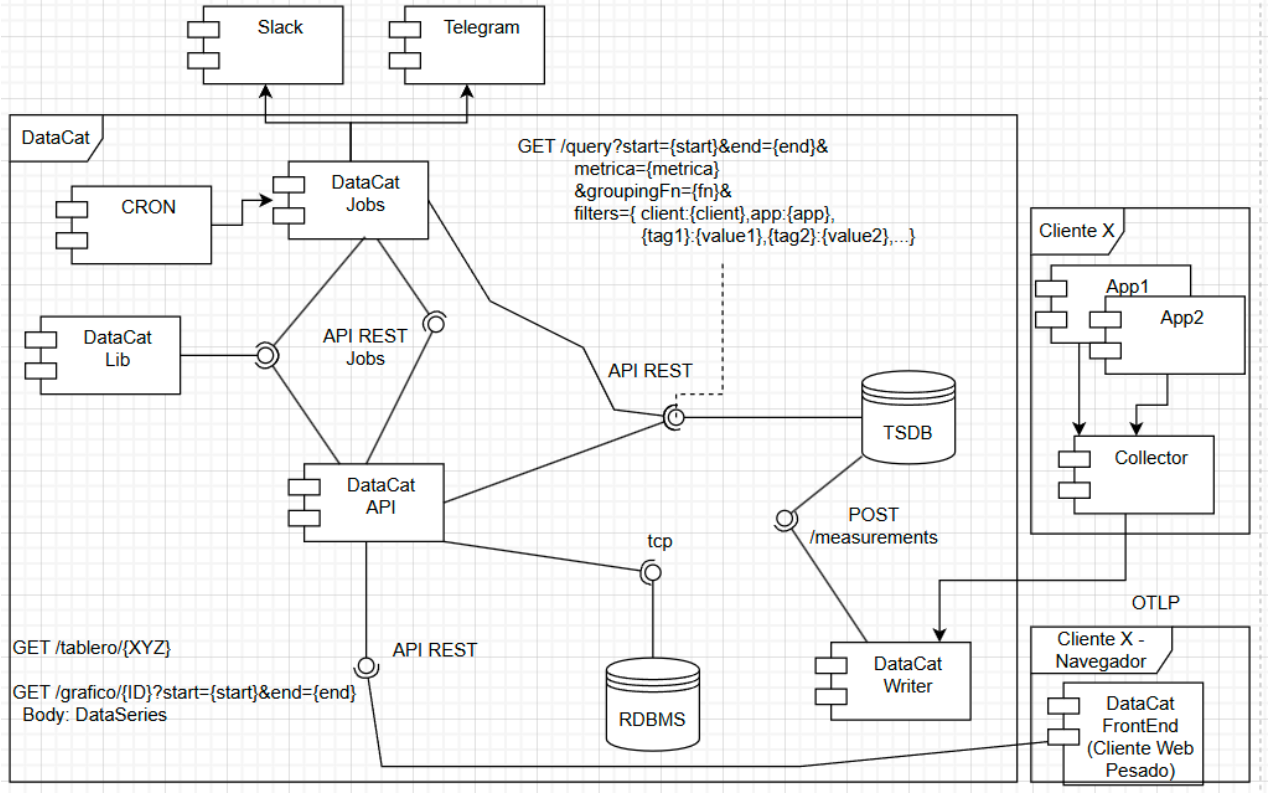
A. Realice los diagramas que considere necesarios para resolver el contexto presentado (clases obligatorias), para las responsabilidades pertinentes a los componentes DataCat Jobs, API y Lib. Indique a qué componente pertenece cada una. **(20 puntos)**
B. Justifique sus decisiones haciendo referencia a los principios de diseño y patrones vistos en clase. **(5 puntos)**
C. **(5 puntos)** Realice un diagrama de objetos, que muestre el tablero XYZ del ejemplo.
D. **(5 puntos)** Realice un diagrama de secuencia, colaboración, o pseudo código que explique como se grafica el componente "ID: 137 **%Error**" del ejemplo.

Punto 3 – Datos (25 puntos)

Diseñar el modelo de datos del componente al sistema para poder persistir en una base de datos relacional a través de un ORM.

A. (**15 Puntos)** Armar la especificación usando un DER físico. Indicando las entidades, sus campos, claves primarias, las foráneas, cardinalidad, modalidad y las restricciones según corresponda.
B. (**10 puntos**) Justificar
    - Qué elementos del modelo es necesario persistir y cuáles no **(atención con esto)**
    - Cómo resolvió los impedance mismatches.
    - Las estructuras de datos que deban ser desnormalizadas, si corresponde.

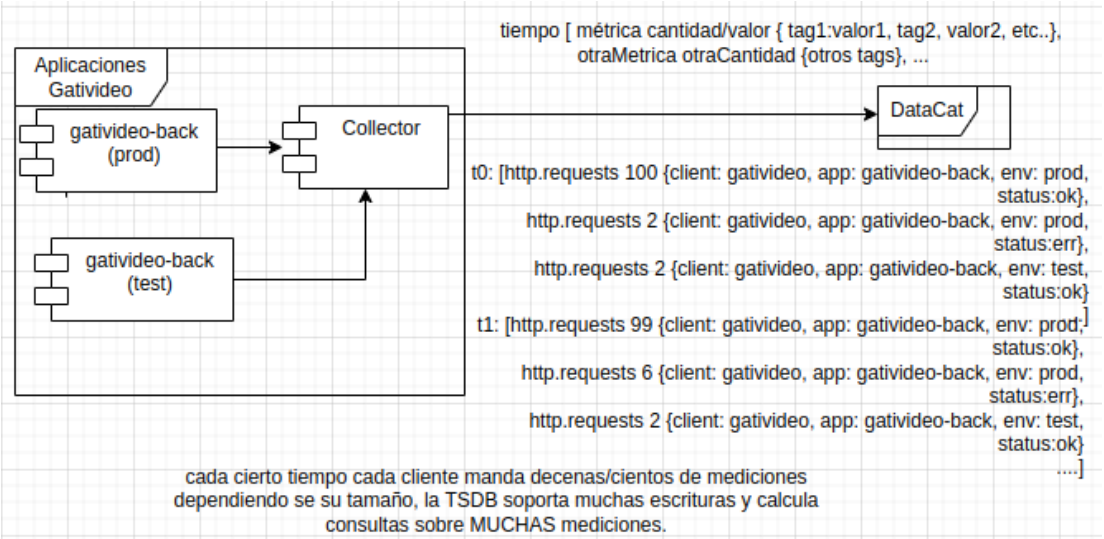


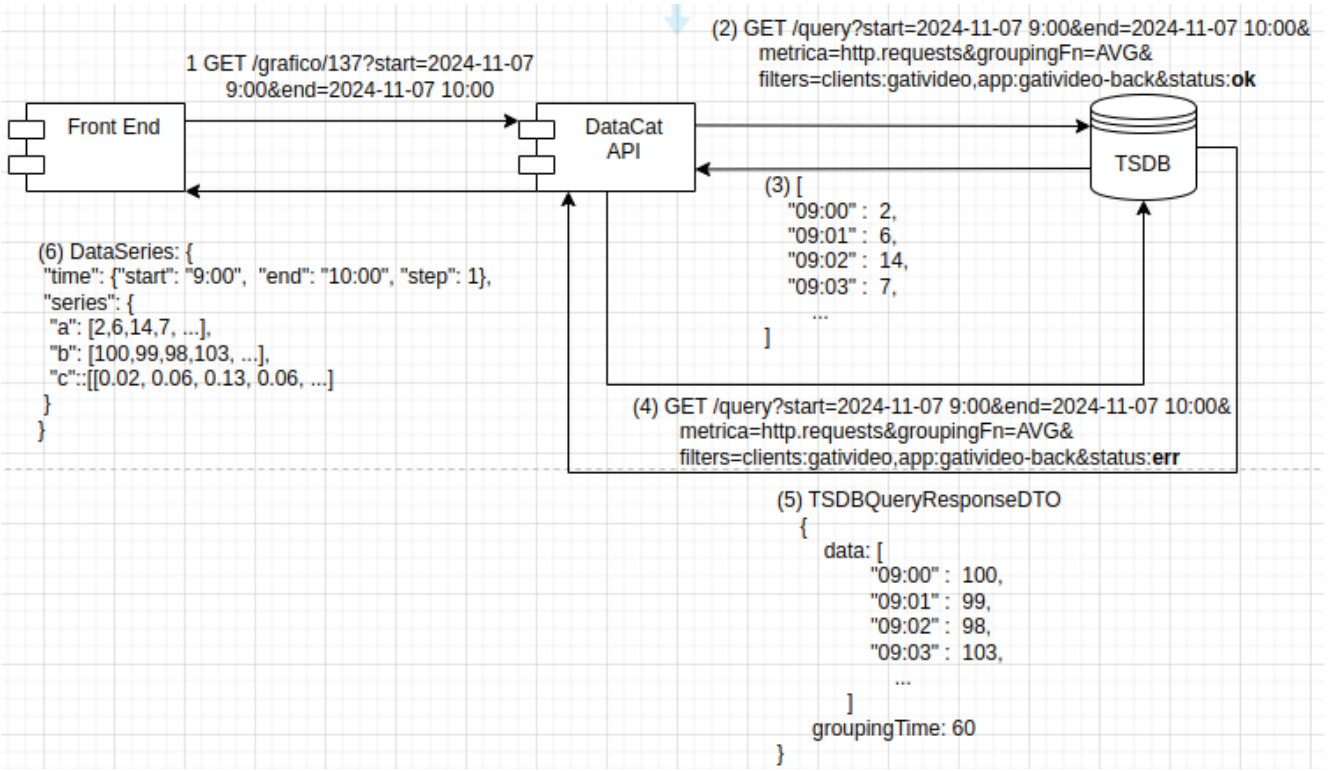


TSDB: Base de datos de series de tiempo, donde se almacenan las métricas / mediciones.
DataCat Writer: se encarga de escribir las mediciones de los clientes. Es el ÚNICO componente que escribe en la misma.
DataCat FrontEnd: cliente web pesado, que se encarga de gestionar las entidades de los clientes y visualizar los tableros. También se encarga de realizar los gráficos, es decir, recibe los resultados de las *consultas* / *fórmulas (Dataseries)* y gráfica lo que corresponde.
DataCat API: backend / gestión de las entidades de los clientes (persistidas en la base relacional - RDBMS) y ejecuta las consultas de los tableros para enviarlas al frontend.
DataCat Jobs: se encarga de ejecutar los monitores, mediante trabajos que se ejecutan cada X tiempo
DataCat Lib: contiene las clases compartidas entre DataCat Jobs y API. Es una LIBRERÍA, no un componente en tiempo de ejecución.

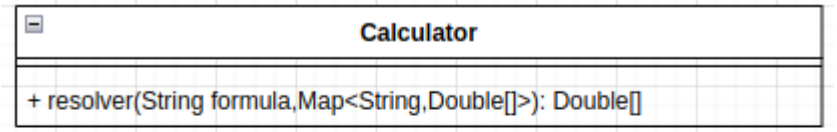
EXAMEN Final - Fecha: 07/12/2023

Apellido, Nombre:.....





Para resolver las fórmulas, cuenta con la clase Calculator.




```
CalculatorTest

assertEquals(
      [0.02, 0.06, 0.13, 0.06],
      resolver(  "a/(a+b)", { "a": [2,6,14,7] , "b": [100,99,98,103] } )
)
```