|  |  |
| --- | --- |
| **Vacante** | **Desarrollador de Plataformas Cloud** |
| **Nombre del candidato:** | Camilo Patiño Vélez |
| **Fecha:** | 28/01/2021 |
| ***Calificación:*** |  |

# **Componente teórico**

1. Usted se encuentra analizando el rendimiento de una aplicación y observa que la distribución de los registros de una tabla DynamoDB entre zonas no es homogénea. Usted descubre que la tabla tiene secciones de llaves primarias nombradas igual y que son usadas constantemente, mientras otras secciones de registros permanecen sin uso.

También, descubre que los códigos postales del usuario son el primer componente de la llave primaria y que la aplicación está siendo mayormente usada por perfiles de dicho código postal. Usted desea cambiar las llaves primarias para que sigan siendo legible para humanos (human readable) y que además permita que las secciones de registros sean homogéneamente distribuidas por DynamoDB. ¿Qué debe hacer?

* 1. Usar valores enteros generados serialmente.
  2. Usar una concatenación de múltiples atributos legibles para humanos.
  3. Usar un subconjunto de un hash MD5 con los contenidos del registro.
  4. Usar una estampa de tiempo UNIX en milisegundos.

1. ¿Cuál es el modelo de datos de DynamoDB?
   1. Tabla, una colección de items. Ítems, con llaves y uno o más atributos. Atributos, con nombre y valor.
   2. Base de datos, una colección de tablas. Tablas, con llaves y uno o más atributos. Atributos, con nombre y valor.
   3. Debido a que DynamoDB no tiene esquema, no hay un modelo de datos.
   4. Ítems, con llaves y uno o más atributos. Atributos, con nombre y valor.
2. La Compañía S.A tiene dos aplicaciones de procesos batch que consumen datos financieros acerca de las transacciones de acciones del día. Cada transacción necesita ser almacenada durable y garantizar que el registro de cada aplicación sea enviado para que el proceso batch de auditoría y facturación pueda procesar los datos. Sin embargo, las dos aplicaciones corren por separado y con muchas horas de diferencia, y necesitan acceso a la misma información de la transacción. Luego de revisar la información de la transacción del día, la información no necesita permanecer almacenada. Cuál es la mejor manera de arquitectar esta solución?
   1. Usar SQS para almacenar los mensajes de las transacciones; cuando el proceso batch de facturación se ejecute primero y consuma el mensaje, escribir el código de una manera que no borre el mensaje luego de ser consumido, de esta manera estará disponible para el proceso de auditoría que correrá horas después. La aplicación de auditoría puede consumir los mensajes de SQS y borrarlos de la cola cuando termine su ejecución.
   2. Almacenar la información de la transacción en una tabla DynamoDB. La aplicación de facturación puede leer los registros mientras que la aplicación de auditoría leerá los registros y luego los borrará.
   3. Usar SQS para almacenar los mensajes de las transacciones. Cuando el proceso batch de facturación consuma cada mensaje, la aplicación creará un mensaje idéntico y lo pondrá en una SQS diferente para que la aplicación de auditoría pueda consumirlo muchas horas después.
   4. Usar Kinesis para almacenar la información de la transacción. La aplicación de facturación consumirá los datos del stream, la aplicación de auditoría puede consumir los mismos datos muchas horas después.
3. Usted está usando lambda (con grandes alocaciones de RAM) para procesar videos cargados a S3 y convertirlos del formato original a H264. Sin embargo, la operación está fallando cuando procesa archivos de video muy grandes. ¿Cuál es la causa de esto?
   1. El tiempo máximo de ejecución que es de 5 minutos por defecto.
   2. Lambda no permite codificación de videos.
   3. Cuellos de botella en la transferencia desde S3.
   4. S3 no se comunica con Lambda.
4. Actualmente, una aplicación acepta que los usuarios carguen archivos en un bucket S3. Usted desea asegurarse de que el nombre de cada archivo cargado se almacene en una tabla de DynamoDB. ¿Cómo se puede lograr esto? Elija 2 respuestas de las opciones que se ofrecen a continuación. Cada respuesta forma parte de la solución.
   1. Cree una función de AWS Lambda para insertar la entrada requerida para cada archivo cargado.
   2. Utilice AWS CloudWatch para sondear cualquier evento de S3.
   3. Agregue un evento con notificación enviada a Lambda.
   4. Agregue el evento de CloudWatch a la sección de secuencias de la tabla de DynamoDB.

# **Componente práctico**

1. **Contexto:**
   1. La empresa Muebles SAS usa un chat en su página web como canal principal de comunicación con sus clientes. Este chat es de un proveedor llamado TuContacto, quien envía varias veces al día información consolidada a Muebles SAS con las estadísticas respecto al motivo de contacto de sus clientes (esta información son la base de la estrategia de fidelización de clientes de Muebles SAS).  
      Actualmente esta información es enviada como un archivo de texto plano a un Bucket de S3 de Muebles SAS.

**Necesidad:**

* 1. Se deben procesar los archivos y realizar las siguientes acciones:
     1. Se debe generar un HASH (MD5) de la información concatenada y validar que sea el mismo HASH que se encuentra en el archivo (así garantizar que los datos son correctos).
     2. Se debe almacenar la información procesada en una tabla de DynamoDB en formato JSON.
     3. Se debe eliminar el archivo plano de S3.
     4. El desarrollo debe contar con pruebas unitarias.

**Contenido del archivo plano:**

|  |
| --- |
| totalContactoClientes=250  motivoReclamo=25  motivoGarantia=10  motivoDuda=100  motivoCompra=100  motivoFelicitaciones=7  motivoCambio=8  hash=2f941516446dce09bc2841da60bf811f |

**El HASH (MD5) se genera a partir de la concatenación de los campos del archivo de la siguiente manera:**

|  |  |
| --- | --- |
| Orden de los campos | totalContactoClientes~motivoReclamo~motivoGarantia~motivoDuda~motivoCompra~motivoFelicitaciones~motivoCambio |
| Ejemplo con los datos anteriores: String base | 250~25~10~100~100~7~8 |
| Ejemplo con los datos anteriores: HASH | 2f941516446dce09bc2841da60bf811f |

**Datos de servicios de AWS:**

* + ARN del bucket:
    - arn:aws:s3:::fdr-developer-test-22122020
    - URI del objeto: s3://fdr-developer-test-22122020/data.txt
  + ARN de la tabla:
    - arn:aws:dynamodb:us-east-1:376141632582:table/fdr-table-22122020
    - Clave principal: “timestamp” (tipo String)

# **Evaluación de competencias**

1. Describa al menos 3 de los 12 factores de aplicación (*The twelve factor app*) los cuales son: Código base (Codebase), Dependencias, Configuraciones, Backing services, Construir - desplegar - ejecutar, Procesos, Asignación de puertos, Concurrencia, Desechabilidad, Paridad en desarrollo y producción, Historiales y Administración de procesos.
2. Describa una situación en la que haya tenido que ir en contra de alguno de los factores listados en el punto anterior e indique sus razones.
3. Usted está en un ambiente ágil trabajando bajo el marco Scrum, y se encuentra con dudas con respecto al requerimiento ¿con quien o quienes considera que debe resolverlos?

Siguiendo la secuencia, primero con el Scrum master y luego con el product owner.

1. ¿Qué es TDD? Y ¿Porque es importante?

Test Driven Development, es desarrollo orientado a pruebas, donde el código escrito es técnico (no legible para cualquier persona) y su sintaxis permite probar directamente las funcionalidades. Es importante porque combina dos partes esenciales del desarrollo en una sola metodología: Las pruebas (unitarias y de integración) y el código mismo, evitando tener un flujo de desarrollo tipo cascada, donde el desarrollo de las pruebas dependan de la finalización del código funcional.