Entrega 1 – Sistema de Conversión Cloud

Daniel Andrés Jiménez Riveros
Andrés Martín Ochoa Toro
Esteban Emmanuel Ortiz Morales
Manuel Felipe Porras Tascón
Desarrollo de Soluciones Cloud
Universidad de los Andes, Bogotá, Colombia

Enlace del Repositorio:
https://github.com/AmOchoat/Cloud-Entrega-1

Enlace Documentación Postman:

https://documenter.getpostman.com/view/5689272/2s93CPrYBi

Contexto del problema:

Una nueva compañía de cloud denominada *Cloud Conversion Tool* desea crear una aplicación web que será ofrecida gratuitamente a usuarios de internet para que estos puedan subir abiertamente diferentes formatos multimedia de archivos y cambiarles su formato o realizar procesos de compresión.

El modelo general de funcionamiento de la aplicación se basa en crear una cuenta en el portal web y acceder al administrador de archivos. Una vez la cuenta ha sido creada, el usuario puede subir archivos y solicitar el cambio de formato de estos para descargarlos. La aplicación web le permite a un usuario convertir archivos multimedia en línea de un formato a otro, seleccionando únicamente el formato destino.

En esta primera versión, la herramienta sólo soporta la compresión de archivos en con tres tipos diferentes de algoritmos y utilidades, estos algoritmos son ZIP, TAR.GZ y TAR.BZ2.

Implementación de la solución:

Para brindar una solución a la compañía *Cloud Conversion Tool,* se realizó la implementación del API REST que permite la creación de una cuenta (*sign up)*, el inicio de sesión(*login*), la creación de una nueva tarea de compresión, el listado de todas las tareas de compresión de un usuario, borrar una tarea de compresión de un usuario y la obtención del archivo original o procesado.

Así mismo se realiza la implementación de una interfaz de usuario que permita consumir el API REST mencionada previamente.

Arquitectura realizada:

A partir de las necesidades manifestadas por el negocio, se definió el siguiente conjunto de diagramas conformado por el diagrama de clases, diagrama de componentes y diagrama de despliegue.

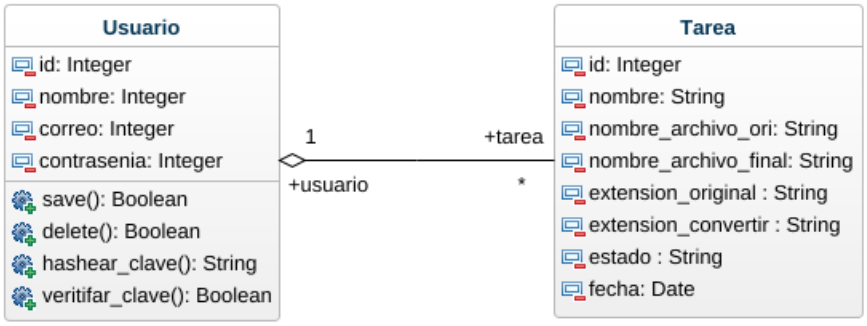

Imagen 1. Diagrama de clases entrega 1 *Cloud Conversion Tool*

En este diagrama se define el usuario y la tarea de conversión. Un usuario tiene muchas tareas de conversión, donde cada tarea tiene un nombre y nombre de los archivos generados, así como la extensión de inicio y final. Lo más importante en este modelo es el estado de una tarea, que puede ser *uploaded* y *processed,* y los nombres de los archivos. Para identificar un archivo en caso de ser necesario, se toma el nombre del archivo asociado a la tarea como valor de texto y se busca este nombre en un directorio de archivos internos.

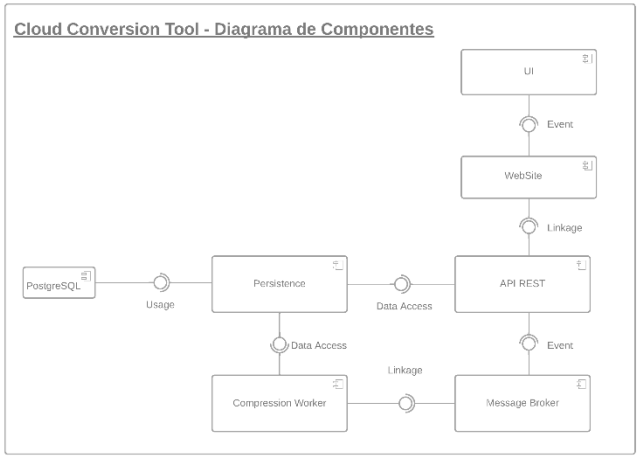

Imagen 2. Diagrama de Componentes entrega 1 *Cloud Conversion Tool*

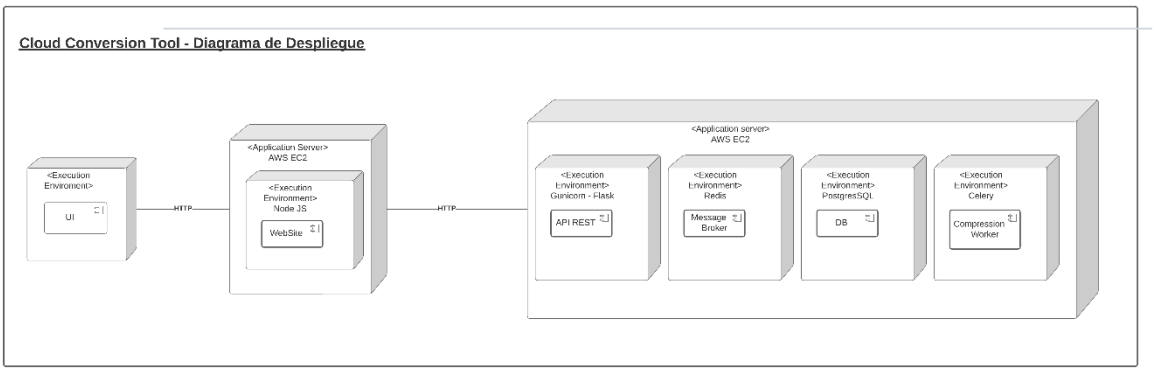

Imagen 3. Diagrama de Despliegue entrega 1 *Cloud Conversion Tool*

En el caso de los diagramas de componentes y diagrama de despliegue se observa la relación de los elementos implementados y los servicios usados en su despliegue. Para el desarrollo de los componentes se hace uso de los servicios requeridos en el curso. En este caso se usa *Flask* para el desarrollo del API, *Redis* y *Celery* para la implementación de la cola de mensajes y demás.

Ventajas y Limitaciones:

Con la arquitectura implementada actualmente, se tiene que el escalamiento del servicio, en caso de necesitar una mayor capacidad de cómputo y almacenamiento, es complicado. Ya que sólo se tienen dos servicios en AWS que se encargan de proveer el servicio, una instancia para la interfaz del usuario y una para el procesamiento y almacenamiento de las solicitudes del usuario, no es posible distribuir las peticiones del usuario en diferentes máquinas virtuales para escalar de manera horizontal la solución. Así mismo, otra limitación existente es la tolerancia a fallos, ya que en caso tal de que alguna de las dos máquinas presente algún fallo, el servicio se dejará de prestar en su totalidad. Esto porque se puede perder el acceso a la interfaz del usuario o la instancia de procesamiento y almacenamiento de datos.

Como contraparte, una primera solución para el escalamiento y procesamiento de las peticiones del usuario es la implementación de una cola de mensajes que permite guardar las solicitudes de los usuarios y procesarlas posteriormente. Esto mejorar de manera parcial el rendimiento y fiabilidad que se le puede dar a las peticiones del usuario.

Trabajo próximo:

Para el mejoramiento del servicio implementado, se puede hacer uso de más servicios en la nube de AWS o similares. Entre estas mejoras que se pueden hacer es generar máquinas virtuales de manera escalable a medida que sean necesarias, así como la implementación de un balanceador de carga entre estas instancias para distribuir el tráfico. Otra consideración que se puede hacer es escalar las máquinas virtuales considerando la cantidad de mensajes en la cola. En caso de que se tenga una cantidad de mensajes mayor a la capacidad de procesamiento actual, se puede aumentar la cantidad de recursos.

Finalmente, en el caso de las bases de datos esta puede ser replanteada con la intensión de ofrecer una redundancia de los datos y aumentar la disponibilidad en caso de fallos, así como guardar los archivos que no se hallan usado en una cantidad determinada de tiempo en una base de datos diferente a los datos que son usados de manera más frecuente.