Universidad de los Andes
Facultad de Ingeniería

Desarrollo de Soluciones Cloud – ISIS 4426,

Proyecto 3.

DOCUMENTACIÓN TÉCNICA

Grupo 5

María Catalina Ibáñez, Jairo Céspedes, David Saavedra, José Manuel Moreno

Contenido

Proyecto 3. 1
Descripción. 3
Requerimientos. 3
Archivos: 4
Diagrama de Arquitectura 5
Base de datos 5
Diagrama de componentes 7
Información de despliegue 9
Conclusiones 9

ILUSTRACIÓN 1. DIAGRAMA DE ARQUITECTURA 5
ILUSTRACIÓN 2. DIAGRAMA DE RELACIONES BASE DE DATOS 7
ILUSTRACIÓN 3. DIAGRAMA DE COMPONENTES 9
ILUSTRACIÓN 4. INFORMACIÓN DE DESPLIEGUE GRUPO DE INSTANCIAS 9
ILUSTRACIÓN 5 INFORMACIÓN DE DESPLIEGUE MÁQUINAS VIRTUALES 9

Descripción.

Enlace del repositorio: https://github.com/Cloud-G05/project1.git
Enlace de la video sustentación: Sustentación proyecto 3 - Cloud (youtube.com)

Este documento detalla la arquitectura técnica implementada en el proyecto 3. Manteniendo la esencia de nuestra segunda entrega, hemos evolucionado la aplicación web para ofrecer un servicio gratuito de conversión de archivos de múltiples formatos (.odt, .docx, .pptx, .xlsx) a PDF. Dicha aplicación destaca por su arquitectura asíncrona, la cual permite la ejecución de tareas en batch sin que los usuarios necesiten esperar en línea. En lugar de ello, se les notifica mediante un cambio de estatus cuando la conversión ha sido completada.

La arquitectura de la solución se basa de servicios en la nube, específicamente a través de Google Cloud Services incluyendo los componentes de backend en dos grupos de instancias auto escalables de 1 a 3 máquinas virtuales que corren el contenedor Docker que contiene el backend. Se tiene dos grupos de instancias porque están ubicados en dos zonas distintas de manera que nuestra aplicación tenga una alta disponibilidad. Por otro lado, el contenedor Docker del worker también corre en un grupo de instancias autoescalables de 1 a 3 máquinas virtuales, mientras que el contenedor Docker del frontend corre en una máquina virtual convencional. Por otro lado, se cuenta con un bucket en Cloud Storage para el almacenamiento de archivos, Pub/Sub para la orquestación de mensajes y Cloud SQL para la base de datos.

Un avance significativo en esta fase es la transición completa a la nube para el despliegue y operación de la aplicación, respondiendo así a una arquitectura nativa de servicios altamente escalables y bajo demanda.

Requerimientos.

Para esta entrega, se ha ampliado el stack tecnológico, incluyendo servicios adicionales de GCP y una arquitectura específica de máquinas virtuales, detallada a continuación:

Load Balancer (LB1): Permite la gestión correcta de cargas o dirige/redirige el tráfico entrante entre los distintos grupos de instancias que alojan las máquinas virtuales donde se ejecutan los contenedores Docker del backend.

Grupo de Instancias 1 (IG1): Incorpora servicios de Compute Engine de GCP que autoescalan de 1 a 3 máquinas virtuales. El primer grupo de instancias aloja el contendor Docker con el backend en la zona us-east4-c.

Grupo de Instancias 2 (IG2): Presta los mismos servicios del grupo de instancias 1 pero se aloja en la zona us-east4-b para garantizar alta disponibilidad.

Máquina Virtual 1 (VM1): Esta máquina virtual corre el contenedor Docker del frontend.

Google Cloud SQL: La gestión de la base de datos se realiza a través del servicio Google Cloud SQL, utilizando PostgreSQL para optimizar el almacenamiento y acceso a los datos.

Google Cloud Pub/Sub: Se ha integrado este servicio para la orquestación de mensajes entre los grupos de instancias que contienen el backend y el grupo de instancias que contienen el worker.

Google Cloud Storage: Se ha integrado este servicio para el almacenamiento de los archivos que se desean convertir y de los archivos convertidos.

Grupo de Instancias 3 (IG3): Este grupo de instancias autoescalable contiene la/s máquina virtual/es que ejecuta/n el contenedor del worker quien a su vez es quien realiza las labores de conversión de archivos. Este grupo de instancias también auto escalan de 1 a 3 máquinas virtuales.

Esta arquitectura propuesta busca no solo cumplir con los requerimientos técnicos del proyecto, sino también ofrecer una plataforma robusta, escalable y de alta disponibilidad para el procesamiento asincrónico de conversión de archivos, facilitando así una mejor experiencia de usuario y un desarrollo sostenible del proyecto en el entorno de la nube.

Archivos:

Este conjunto de diagramas detalla la arquitectura de la aplicación, destacando tanto la estructura de contenedores Docker como la interacción de los componentes de software.

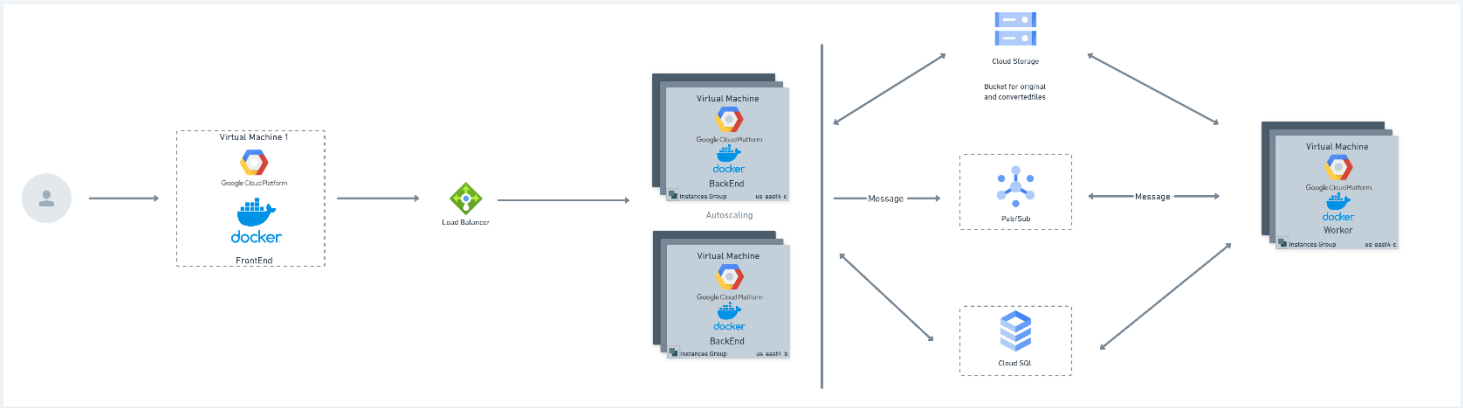

Ilustración 1. Diagrama de Arquitectura

Diagrama de Arquitectura

En contraste con entregas previas, los componentes hacen parte de servicios nativos de la nube, específicamente de Google Cloud Services.

Base de datos

La base de datos mantiene la misma estructura construida para la entrega 1, la misma se detalla a continuación:

Entidades Principales:

1. **User (Usuario):**
    - **username: str** - El nombre de usuario, el cual es único para cada usuario.
    - **email: str** - La dirección de correo electrónico del usuario, utilizado como identificador único para cada usuario.
    - **password: str** - La contraseña del usuario para autenticación.

Cada usuario puede tener asociadas múltiples tareas de conversión de archivos, lo que refleja una relación uno a muchos con la entidad Task.

2. **Task (Tarea):**
    - **id: str** - Un identificador único para cada tarea de conversión.
    - **name: str** - El nombre o título asignado a la tarea de conversión.

- **original_file_ext: str** - La extensión del archivo original antes de la conversión.
- **file_conversion_ext: str** - La extensión deseada del archivo después de la conversión.
- **available: bool** - Un valor booleano que indica si el archivo convertido está disponible para descarga.
- **status: enum** - Un enumerado que representa el estado de la tarea (por ejemplo, 'UPLOADED' para tareas cargadas y 'PROCESSED' para tareas cuya conversión ha finalizado).
- **time_stamp: datetime** - La fecha y hora en que se creó la tarea.
- **input_file_path: str** - La ruta de acceso al archivo original almacenado.
- **output_file_path: str** - La ruta de acceso al archivo convertido almacenado.

La entidad Task está vinculada a la entidad User, indicando que cada tarea está asociada a un usuario específico.

Relaciones:

- **Relación Usuario-Tarea:** Cada usuario puede tener varias tareas asociadas, pero cada tarea está vinculada a un solo usuario. Esto establece una relación uno a muchos entre User y Task, representada en el diagrama.

Enumeraciones:

- **status: enum** - Define los posibles estados de una tarea con los valores 'UPLOADED' (indicando que el archivo ha sido cargado, pero no procesado aún) y 'PROCESSED' (indicando que el archivo ha sido procesado y está listo para ser descargado).

Diagrama de relaciones

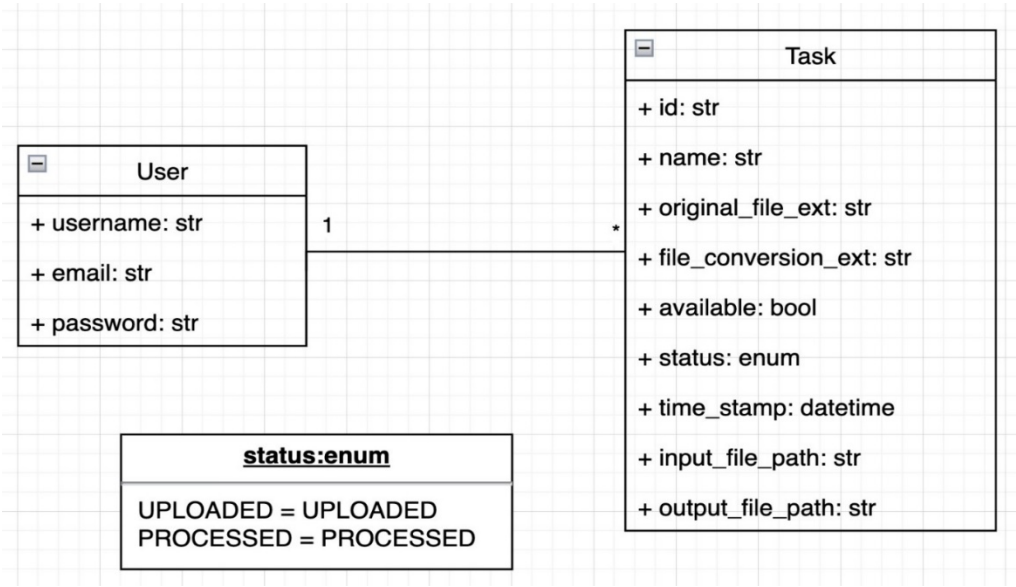

Ilustración 2. Diagrama de relaciones base de datos

No obstante, vale la pena reiterar que dicha estructura de datos ahora se encuentra alojada en el servicio de Google Cloud SQL con PostgreSQL tal como se indicó anteriormente.

Diagrama de componentes

De igual manera, los componentes utilizados mantienen la misma estructura. El diagrama de componentes proporciona una visión clara de la arquitectura de la solución implementada para el proyecto. Este esquema refleja la interacción entre los distintos componentes de la aplicación, asegurando una operación eficiente y alineada con los estándares actuales de la industria para soluciones web escalables.

Componentes Principales:

1. **Usuario (User)**:
   - Autenticación mediante **JSON Web Tokens (JWT)**: Asegura una comunicación segura y verifica la identidad del usuario al interactuar con la plataforma.
2. **Frontend (React.js)**:
   - Proporciona una interfaz de usuario interactiva y dinámica, desarrollada con el framework moderno de JavaScript, React.js.
   - Envía solicitudes HTTP al backend, autenticadas mediante JWT y maneja las respuestas para reflejar los cambios en la interfaz de usuario.
3. **Backend (FastAPI)**:

- Gestiona las solicitudes HTTP provenientes del frontend, procesando la lógica de negocio y las operaciones de la aplicación.
- Interactúa con la base de datos PostgreSQL para recuperar o almacenar información, con Cloud Pub/Sub para publicar tareas por hacer en forma de mensaje y con Cloud Storage para acceder y almacenar archivos.
- Envía y recibe datos en formato JSON, asegurando un intercambio de datos eficiente y estandarizado.

4. **Worker (Python):**
   - Realiza la conversión de archivos de un formato dado (.odt, .docx, .pptx, .xlsx) a pdf.
   - Extrae mensajes de Cloud Pub/Sub que le informan de una nueva tarea de conversión de archivo.
   - Se comunica con la base de datos para actualizar el estado de las tareas, asegurando que los usuarios puedan rastrear el progreso de sus conversiones.
   - Interactúa con Cloud Storage para acceder y almacenar archivos en un bucket.
5. **Base de Datos (PostgreSQL)**:
   - Almacena y gestiona los datos de la aplicación, incluyendo información de usuarios y detalles de las tareas de conversión.
   - Recibe consultas del backend y actualiza el estado de las tareas a medida que el worker convierte archivos (realiza tareas).
6. **Almacenamiento de archivos (File Storage):**
   - Almacenamiento de archivos originales cargados por el usuario en un bucket de Cloud Storage.
   - Almacenamiento de archivos convertidos en un bucket de Cloud Storage.
7. **Sistema de mensajería (Pub/Sub):**
   - Almacena los mensajes que publica el backend.
   - Permite, mediante una suscripción a un tema, que el worker acceda a estos mensajes publicados por el backend.

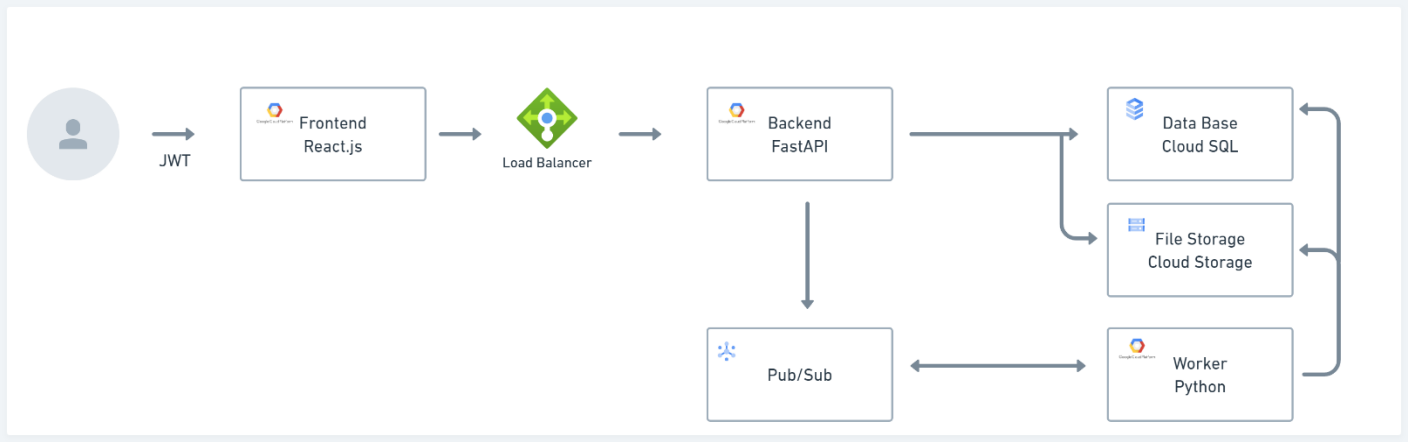

Ilustración 3. Diagrama de componentes

Información de despliegue

La información de las máquinas virtuales desplegadas en la nube es fácilmente identificable en la ilustración 4 y 5 que contienen a su vez la información más relevante sobre los grupos de instancias auto escalables, así como de las máquinas virtuales en sí mismas.

Estado	Nombre	Instancias	Plantilla	Tipo de grupo	Fecha y hora de creación	Recomendación	Ajuste de escala automático	Zona	En uso por
✓	web-server-group	1	web-server-template-v9 (Regional)	Administrado	abr 19, 2024, 10:37:33 a. m. UTC-05:00		Activado: Objetivo Uso de CPU 60 %	us-east4-c	http-backend
✓	web-server-group2	1	web-server-template-v9 (Regional)	Administrado	abr 23, 2024, 5:46:00 p. m. UTC-05:00		Activado: Objetivo Uso de CPU 60 %	us-east4-b	http-backend
✓	worker-instance-group	1	worker-template-v2 (Regional)	Administrado	abr 23, 2024, 4:19:16 p. m. UTC-05:00		Activado: Objetivo Uso de CPU 55 %	us-east4-c	

Ilustración 4. Información de despliegue grupo de instancias

Adicionalmente, se evidencian las máquinas virtuales desplegadas previamente y que no se encuentran activas. De igual manera, es necesario tener en cuenta que dichas máquinas se despliegan únicamente con propósitos de uso y testeo procurando disminuir los costos por uso excesivo.

Estado	Nombre	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar
○	web-server	us-east4-c			10.150.0.3 (nic0)		SSH
✓	web-server-group-n1pt	us-east4-c		web-server-	10.150.0.48 (nic0)	35.194.79.11 (nic0)	SSH
✓	web-server-group2-8txf	us-east4-b		web-server-	10.150.0.49 (nic0)	35.236.194.46 (nic0)	SSH
○	worker	us-east4-c			10.150.0.4 (nic0)		SSH
✓	worker-instance-group-357p	us-east4-c		worker-	10.150.0.47 (nic0)	35.245.141.176 (nic0)	SSH

Ilustración 5 Información de despliegue máquinas virtuales

Conclusiones

Al evaluar la evolución y las adaptaciones realizadas para la segunda entrega del proyecto, resulta evidente que la inclusión de una arquitectura basada en la nube ha marcado un hito significativo hacia la consecución de una mayor escalabilidad y sostenibilidad del producto. Este cambio estratégico no solo ha inyectado una mayor flexibilidad y robustez al sistema, sino

que también ha realzado su capacidad para adaptarse a demandas fluctuantes, un aspecto crítico para la viabilidad a largo plazo del proyecto.

Sin embargo, la transición hacia servicios en la nube ha introducido desafíos notables, especialmente en lo que respecta al manejo eficiente del presupuesto. La necesidad de equilibrar costos con la operatividad del sistema se ha convertido en una variable crítica, especialmente al contrastar las limitaciones y disponibilidad de los servicios en la nube en comparación con la infraestructura local previamente empleada. La gestión de recursos en la nube, por tanto, requiere de un escrutinio y planificación detallados para evitar escaladas de costos no previstas.

En este contexto, se identifica como fundamental la exploración y posible integración de servicios de escalado automático que puedan ajustar los recursos necesarios en tiempo real según la demanda, optimizando así tanto el rendimiento como los costos asociados. Igualmente, es imperativo evaluar continuamente nuevos servicios y herramientas en la nube que ofrezcan alternativas más coste-efectivas y flexibles para la gestión de recursos, permitiendo así un control más directo y eficiente sobre el presupuesto sin sacrificar la calidad ni la disponibilidad del servicio ofrecido.