Universidad de los Andes
Facultad de Ingeniería

Desarrollo de Soluciones Cloud – ISIS 4426,

Proyecto 1, entrega 2.

DOCUMENTACIÓN TÉCNICA

Grupo 5

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

Contenido

Universidad de los Andes
Facultad de Ingeniería

Descripción.

Enlace del repositorio: https://github.com/Cloud-G05/project1.git
Enlace de la video sustentación: https://youtu.be/58Ea-bwE1oM

Este documento detalla la arquitectura técnica implementada en la segunda fase del Proyecto 1. Manteniendo la esencia de nuestra primera 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 centra en un backend construido con Python y utiliza FastAPI para el desarrollo de una API REST, PostgreSQL para la gestión de bases de datos, Celery para la orquestación de tareas en batch con Redis como sistema de mensajería (broker), y React para el frontend. Un avance significativo en esta fase es la adopción de componentes en la nube para el despliegue y operación de la aplicación con un enfoque particular en los servicios de Google Cloud Platform (GCP). Esto se ha hecho con el objetivo de mejorar la escalabilidad y disponibilidad del proyecto.

Requerimientos.

Para esta entrega, se ha ampliado el stack tecnológico incluyendo servicios adicionales de GCP y una arquitectura específica de máquinas virtuales usando los servicios de Compute Engine. Todas las máquinas virtuales son de tipo e2-small con 20GB de almacenamiento y usan la imagen de Ubuntu 20.04. A continuación se detalla la arquitectura:

Máquina Virtual 1 (VM1): La primera máquina virtual corre dos contenedores Docker, el del backend y el del frontend.

Máquina Virtual 2 (VM2): Esta máquina virtual hospeda y corre el contenedor Docker que ejecuta la cola de tareas de Celery. Adicionalmente, en esta máquina se corre Redis; no obstante, el bróker de mensajería se corre directamente sobre la VM.

Máquina Virtual 3 (VM3): Designada para albergar el servidor de archivos, es decir, es la que almacena los archivos originales y convertidos. Esta máquina virtual funciona como servidor NFS, mientras que las VM1 y VM2 actúan como clientes NFS.

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.

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 conversiones 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.

Diagrama de Arquitectura

En contraste con la primera entrega, se ha migrado la operación de los contenedores Docker de un ambiente local a los servicios en la nube de GCP.

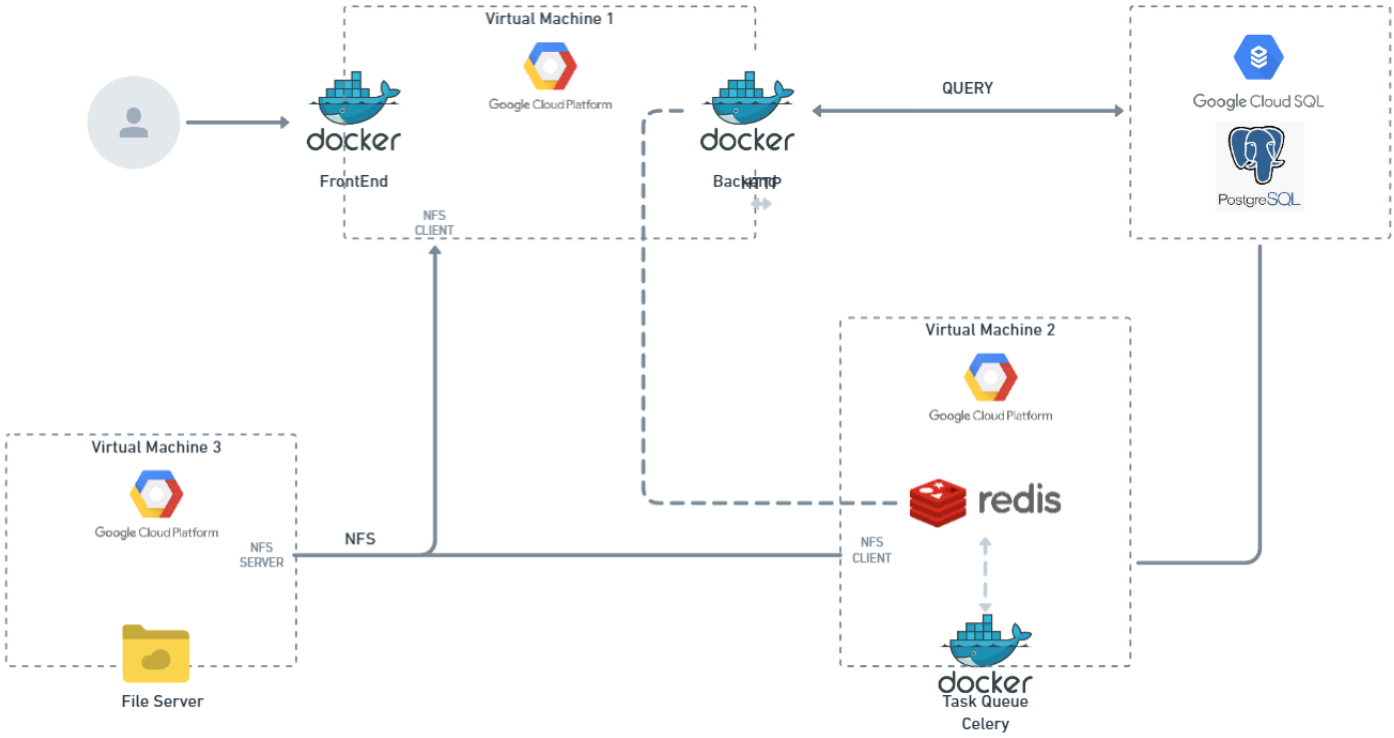

Ilustración 1. Diagrama de Arquitectura

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 original y 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

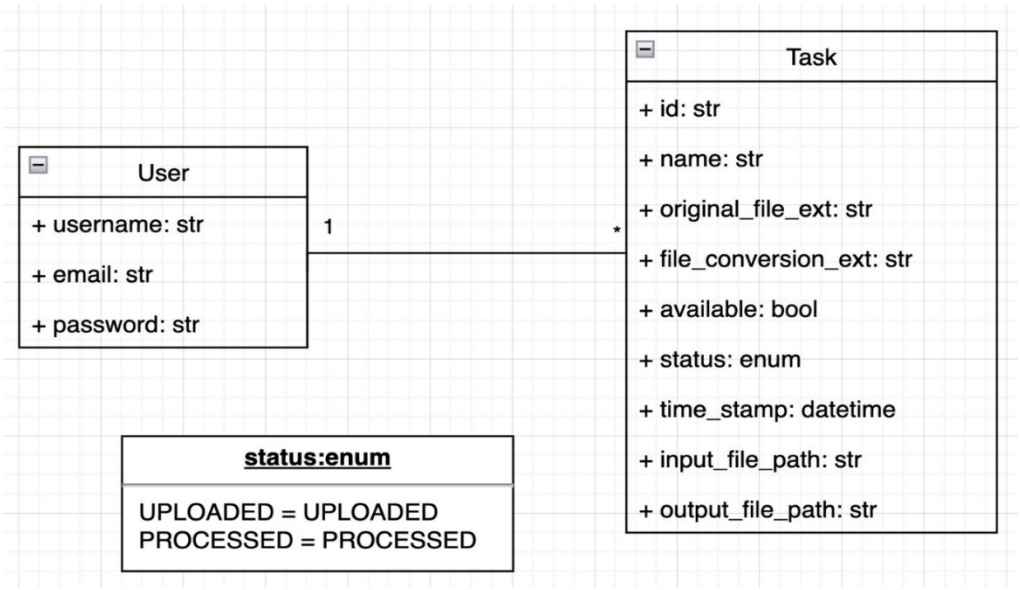

Ilustración 2. Diagrama de relaciones base de datos

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.

Sin embargo, con la nueva infraestructura planteada, la comunicación entre Redis y el backend se realiza directamente entre el contenedor Docker del backend alojado en la Máquina Virtual 1 y la Máquina Virtual 2 que corre a Redis.

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 y con la cola de tareas para la gestión de procesos asíncronos.
    - Envía y recibe datos en formato JSON, asegurando un intercambio de datos eficiente y estandarizado.
4. **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 según los cambios en la cola de tareas (TaskQueue).
5. **Cola de Tareas (Celery)**:
    - Maneja procesos asíncronos, como las tareas de conversión de archivos, permitiendo que el backend responda a solicitudes de usuario sin retrasos.
    - 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.
6. **Broker de Mensajes (Redis)**:
    - Actúa como intermediario entre el backend y Celery, facilitando la distribución y gestión de tareas.
    - Permite una comunicación eficaz y la sincronización del estado entre la base de datos y la cola de tareas.

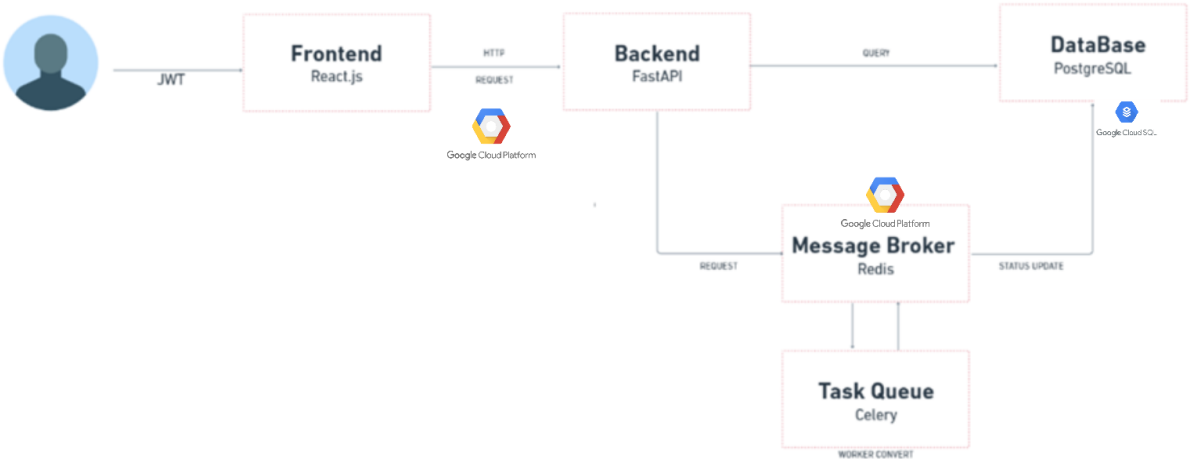

Ilustración 3. Diagrama de componentes

Información de despliegue

La información de las maquinas virtuales desplegadas en la nube es fácilmente identificable en la ilustración 4 que contiene a su vez la información más relevante sobre las tres máquinas virtuales desplegadas.

Estado	Nombre	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar
✓	file-server	us-east4-c	Ahorrar $9/mes		10.150.0.2 (nic0)	34.48.78.165 (nic0)	SSH
✓	web-server	us-east4-c			10.150.0.3 (nic0)	35.245.238.207 (nic0)	SSH
✓	worker	us-east4-c			10.150.0.4 (nic0)	34.85.168.198 (nic0)	SSH

Ilustración 4. Información de despliegue

Es necesario tener en cuenta que dichas máquinas se despliegan únicamente con propósitos de uso y testeo procurando disminuir los costos por uso excesivo.

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.